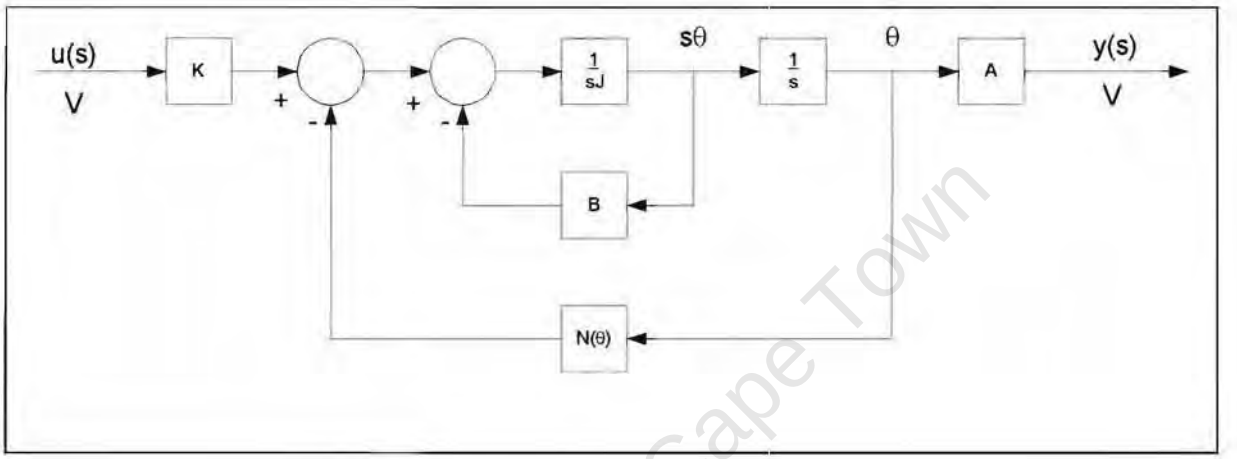
The robustness of H-infinity control



Prepared By

Fausto Marques
Postgraduate control engineering student
Department of Electrical and Electronic engineering
University on Cape Town

Prepared For

Department of Electrical and Electronic Engineering
University of Cape Town
South Africa

30 September 1999

This dissertation is being submitted as complete fulfillment of a Master's degree in electrical engineering

Acknowledgments

I would like to thank my supervisor, Professor Martin Braae, of the Department of Electrical and Electronic Engineering at the University of Cape Town, South Africa, for all his support and insight during the course of this project.

Terms of Reference

This masters dissertation was supervised by Professor Martin Braae, who also gave the briefing on the material to be covered.

The initial briefing took place in April 1998. The purpose of the briefing was to clarify the requirements of the dissertation and to provide insight into a research project. Further meetings were held on a weekly basis to judge progress and discuss any major findings.

Professor Braae's instructions were as follows:

- To investigate the paper entitled "Robust, Fragile or Optimal" written by Keel et al (1997)
- To apply their theory to an open loop unstable plant and thus provide judgement on what was presented. The plant chosen was a non-linear second order robot arm plant
- To design an H-infinity controller for the robot arm plant and apply parametric stability margin analysis
- To implement the controller on the physical plant to judge its performance
- To draw conclusions from this practical performance and therefore provide comment on the paper by Keel et al

The dissertation would be due on 30 September 1999.

Synopsis

Modern control theory generates controllers of a high order. Since these controllers inherently require elaborate circuits or algorithms for implementation, there is always the possibility that the implemented controller will differ from the designed controller by a certain degree. Furthermore, the control engineer might want to tweak the controller in practice and will therefore deliberately adjust the parameters of the nominal controller. The key factor is that the controller although perturbed from the nominal controller, will still stabilize the closed loop system. The greater this perturbation can be without destabilizing the closed loop system, the more robust the controller is.

Keel et al, in their paper entitled "Robust, Fragile or Optimal?" (1997), made a statement that the controllers generated by H-infinity design methods are fragile. A norm was introduced called the parametric stability margin, to serve as a measure of this robustness. A fragile controller is defined as a controller that is very sensitive to changes in its controller coefficients and any small change from the nominal controller will result in closed loop instability. This type of controller will have a very small parametric stability margin. This parametric stability margin is defined as a radius in parameter space in which the controller will be stable in closed loop. If the norm of the perturbation exceeds this margin in parameter space, then the closed loop system will become unstable.

A real plant was chosen as a means to test the claims of Keel et al. The plant is a simple "robot arm", a non-linear second order system. The non-linearity creates both open loop stable and unstable regions for control. A controller was designed for this plant using H-infinity techniques. This controller would form the basis for testing the claims of Keel et al. When this controller was analysed using the parametric stability margin, it was predicted to be fragile or stated differently: very small changes in the controller coefficients would destabilize the closed loop system. However, closer scrutiny revealed that this sensitivity was only concentrated on the leading coefficients in the numerator and denominator of the controller. Furthermore, the relative size of the perturbations on these coefficients was far in excess of 1000% of the original coefficient.

The designed controller was implemented successfully in practice using a digital implementation. Even a perturbed version of 200% of the controller coefficients stabilized the closed loop system. It was then discovered that it was possible to create a perturbation with a norm greater than the parametric stability margin that would still stabilize the closed loop system. A similar perturbation could also be constructed for the examples presented by Keel et al in their paper (1997).

The resulting conclusion was that the H-infinity techniques actually generate rather robust controllers. Provided that the perturbations on the leading controller coefficients are kept below the destabilizing value, the other coefficients can be perturbed to a very large degree. This destabilizing value is given by the perturbation vector at the parametric stability margin. This perturbation will place some closed loop poles on the stability boundary of the region of interest. In this case, the stability region is the left half s-plane. The dominant parameters in this vector are identified as those that have the largest percentage perturbation relative to the original coefficient. In most cases, this turns out to be the leading coefficients. A more accurate definition of fragility should therefore be whether the maximum relative percentage perturbation for destabilization, is less than a certain percentage.

The results of this dissertation therefore make the control engineer wary of the fact that the results obtained from analysis using the parametric stability margin are not necessarily conclusive. Well-designed controllers might be predicted to be fragile. Before accepting this verdict, the control engineer is encourage to scrutinize the results before discarding a controller that might actually work very well in practice.

Table of Contents

List of Figures

Nomenclature

General

G	Matrix or vector G is indicated in bold
$\|\cdot\|_\infty$	H-infinity norm operator
$\|\cdot\|_2$	L_2 norm operator
$[a \quad b \quad c]^T$	Transpose of a vector or matrix

Set theory and spaces

S	Stability region in complex space
C	Complex space
∂S	Boundary of S
U	The exterior of the stable region, U = C – S
U^0	The interior of U, U^0 = C – S - ∂S

These sets are related as follows:

$S \cup \partial S \cup U^0 = C$ and $S \cap U^0 = S \cap \partial S = \partial S \cap U^0 = \varnothing$

1 Introduction

Background

Modern control theory has introduced design methods that tend to create very high order controllers. These design methods include H-infinity, LQG (Linear Quadratic Gaussian) and mu-synthesis. These high order controllers are more difficult to implement in practice and often require very high precision electronics. As a result of this, the designer needs to spend a lot of time to match the coefficients as closely to the designed controller as possible. Even digital implementation has its flaws due to the fixed point arithmetic and rounding errors on personal computers. For this reason, the controllers should be robust to changes in the coefficients in order to tolerate inaccuracies in their implementation. If the controller is not robust, then the controlled system will be destabilized for small perturbations in the controller coefficients. This problem of controller sensitivity is thus a very important consideration in modern design.

Furthermore, the design methods do not directly take controller uncertainty into account. They mainly concentrate on stabilizing a system for a certain degree of plant uncertainty. It is then assumed that the designed controller will be implemented exactly in practice. However, as this assumption is not always possible, well-designed controllers in theory will not necessarily work in practice. The problem of controller sensitivity is thus an important issue to address.

This sensitivity or fragility was addressed by L.H. Keel and S. P. Bhattacharyya in their paper entitled "Robust, Fragile or Optimal ?". This article appeared in the IEEE Transactions on Automatic Control in 1997. Keel and Bhattacharyya presented a number of quoted design problems and analyzed them using a *parametric stability margin.* This margin served to indicate that the controllers designed in these examples were in fact so sensitive to changes in their coefficients, that they would be unusable in industry.

In order to verify these claims, it was decided to perform their analysis on a physical plant of the same form as the quoted examples. The plant chosen in this dissertation is a second order *robot arm plant.*

Objectives

The objectives of this dissertation are as follows:

- To investigate H-infinity design in order to design a controller
- To derive a model for the robot arm plant
- To produce a controller, through simulation, for the robot arm plant using H-infinity design
- To analyze the work presented by Keel and Bhattacharyya in their paper
- To apply Keel and Bhattacharyya's theory and analysis to the controlled robot arm plant
- To implement the controller in practice and draw conclusions from its performance in relation to the theoretical predictions

Procedure

The results obtained have been achieved through mathematical work and digital simulation using an empirical approach. The design of the controller for the robot arm plant was a very iterative process whereby different characteristics were continually tweaked to generate a decent design for practical implementation. The designs and mathematical calculations were implemented using Matlab together with the robust, control and symbolic toolboxes. A simulator was also written in Visual C++ in order to implement a digital controller on the physical plant.

Limitations

The study was limited to a single plant that should have exhibited very similar characteristics to those quoted by Keel and Bhattacharyya. However, as shall be seen, the results ended up differing quite drastically from those obtained by Keel and Bhattacharrya. As a result, a second simpler theoretical example was chosen for analysis. The scope of the theory investigated was also largely limited to that presented by Keel and Bhattacharyya in their paper.

Plan of development

This dissertation begins by introducing the parameter uncertainty in control system design and then highlights the aspect of controller sensitivity. A proposed means of identifying very sensitive controllers is then introduced as the parametric stability margin. These factors constitute the problem statement. The dissertation then provides an overview of H-infinity design theory to form a basis for the controller design for the robot arm plant. The next section covers the robot arm plant as well as its controller design. The section also considers the physical implementation of the robot arm controller and its success in practice. This is followed by a presentation of the L_2 parametric stability theory and its calculation. The next section is the stability analysis of the robot arm plant, using the parametric stability margin. Finally, the conclusions and recommendations are given.

2 Problem Statement

2.1 A generalized SISO control loop and parameter uncertainty

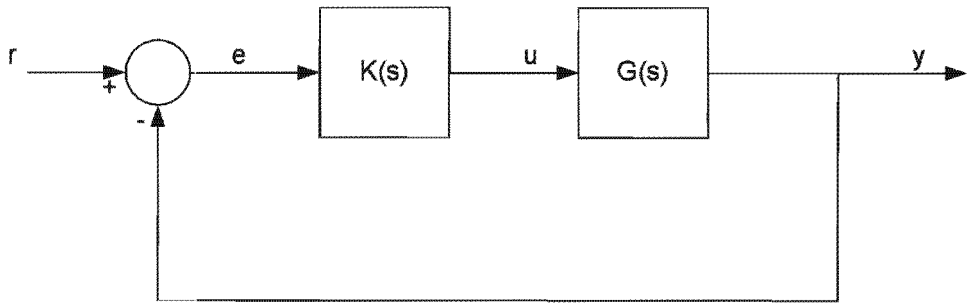


Figure 2.1 A unity feedback SISO control loop

Figure 2.1 shows a unity feedback SISO loop. The controller K(s) is designed to stabilize the system in closed loop. Most design techniques design controllers that compensate for a certain measure of plant uncertainty. Consider that the controller K(s) stabilizes the nominal plant $G_0(s)$. The designed controller should also stabilize a family of plants around this nominal plant. This implies that K(s) should stabilize all plants of the form $G_0(s)+\Delta G(s)$ where $\Delta G(s)$ is limited to a certain size. What most design methods do not address is the uncertainty in the controller. For example, K(s) might stabilize the family of plants $G_0(s)+\Delta G(s)$. However, if K(s) was actually implemented as $K(s)+\Delta K(s)$, then it is very possible that the closed loop system would be destabilized. It is this controller sensitivity that is considered in this dissertation.

2.2 Parameter uncertainty under H-infinity

The core of H-infinity design is to minimize the H-infinity norm of certain cost functions. The uncertainties presented in the previous section can be quantified by making use of this H-infinity norm.

Consider the family of plants:

$$G(s) = G_0(s) + \Delta G(s)$$
$$\text{where } \left\|\Delta G(s)\right\|_\infty \leq \alpha$$

This specification implies that the family of plants is limited to a prescribed ball in parameter space of radius α.

An additional constraint on the system, that is not included in design methods, is the uncertainty on the controller itself.

Consider the family of controllers:

$$K(s) = K_0(s) + \Delta K(s)$$
$$\text{where } \|\Delta K(s)\|_\infty \leq \alpha$$

This controller will stabilize the family of plants provided that the perturbation from the nominal controller does not exceed α in norm space. If α can be made very large while still stabilizing the closed loop system, then the controller is robust. However, if α is very small, then the controller can be labelled as fragile, as any small deviation from the nominal controller destabilizes the closed loop system.

2.3 L_2 parametric stability margin

The points raised above are very general. Keel et al analyzed their quoted examples using a similar procedure. However, their analysis was from a parametric point of view. Here the perturbations are considered per coefficient in the controller. The norm of this perturbation then gave an indication as to how robust the controller was to changes in its coefficients. Although the H-infinity norm could have been used as the measure, Keel et al made use of the L_2 norm.

2.4 The controller sensitivity problem

The factors mentioned in sections 2.1 to 2.3 form part of what can be called a controller sensitivity problem. This issue is often avoided in modern controller design as it is assumed that controllers are implemented exactly the same in practice as they are expressed in theory.

Referring to section 2.2 the following analogy can be made: the nominal controller is the controller designed electronically using a computer-aided design. The perturbed controller is the controller that is implemented in practice. Owing to a number of factors such as fixed-point arithmetic and round-off errors and inaccuracies in components used to implement the controller, the perturbed controller can differ quite a lot from the nominal controller. If the controller is fragile, then this difference is bound to destabilize the closed loop system.

Keel et al raised this issue in their paper regarding controllers that are generated by H-infinity techniques (1997). A calculation was presented called the L_2 parametric stability margin as a measure of this fragility or robustness. This dissertation will investigate their claims by considering how absolute this measure is in classifying the fragility of a controller.

3 H-infinity design

3.1 An Overview of H-infinity Theory

H-infinity forms part of optimal control design methods. The basis of H-infinity is to minimize the H-infinity norm of a cost function. Consider a standard representation of an uncertain plant under feedback control. This is shown in Figure 3.1 below.

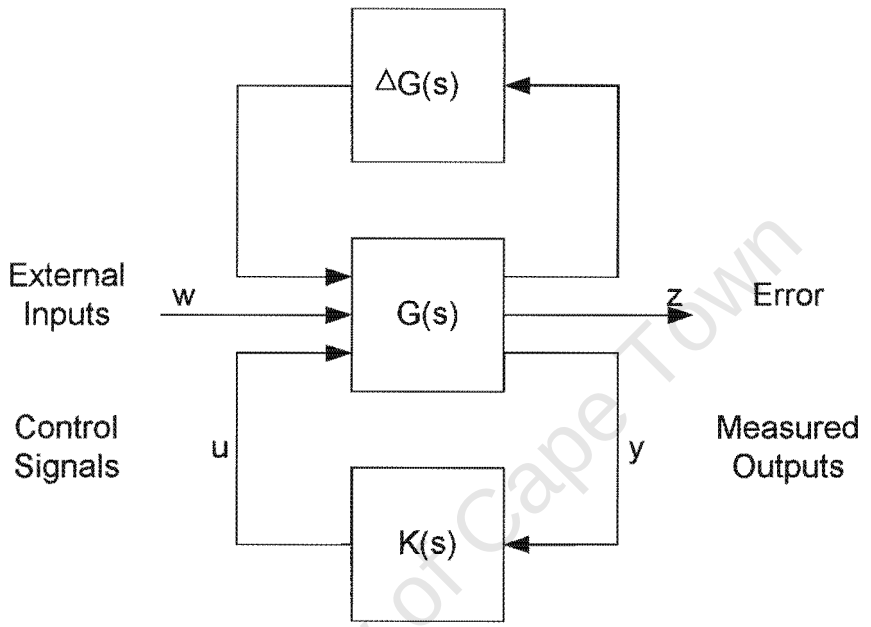


Figure 3.1 Standard representation of an uncertain plant under feedback control

The relationship between w and z can be expressed as

$$z = \left[G_{11} + G_{12}K\left(I - G_{22}K\right)^{-1}G_{21}\right]w$$

which is often rewritten as

$$z = F_1\left(G, K\right)w$$

A detailed derivation for the above can be found in Appendix 9.1.1.

The H-infinity optimization problem is to minimize the H-infinity norm of F_1 over all realizable controllers K(s) that stabilize the closed loop system. Refer to Appendix 9.1.2 for a description of the H-infinity norm.

Many problems can be placed into a form suitable for H-infinity optimization. For example, the characteristic in question could be the sensitivity function for the closed loop system. In this case the constraint could be expressed as follows:

$$\text{minimize } \|S\|_\infty$$
$$\text{where } S = (I + GK)^{-1}$$

By making use of the Youla Parameterization of all stabilizing controllers, it is possible to formulate the design as an unconstrained optimization problem.

The Youla Parameterization makes the following substitution:

$$\text{Set } Q = K(I + GK)^{-1}$$

Then the minimization problem can be expressed as follows:

$$\underset{\text{stable Q}}{\text{minimize}} \|W(I - GQ)\|_\infty \qquad \text{(See Appendix 9.1.3)}$$

The problem is unconstrained in that the only constraint is that Q be stable and proper.

The cost functions for two different H-infinity problems are shown below.

Sensitivity minimization

$$F_1(P, K) = W(I + GK)^{-1}$$

where W is a frequency dependent weighting function on the sensitivity function.

This optimization problem ensures good setpoint tracking with no constraints placed on disturbance rejection.

Mixed Performance and Robustness objective

$$F_1(P,K)=\begin{bmatrix} W_1 S \\ W_2(I-S) \end{bmatrix}$$

where W_1 is a frequency dependent weighting function on the sensitivity function
and W_2 is a frequency dependent weighting function on the complementary sensitivity function.

This optimization problem ensures good setpoint tracking as well as good disturbance rejection. The disturbance rejection is ensured by keep (I-S) small in magnitude. As the sum of the sensitivity and complementary sensitivity functions is equal to 1, the designer is forced to minimize S over a certain range and (I-S) over a different range. Both functions cannot be minimized at the same point otherwise the constraint on their sum will be broken. For this reason the W_1 weighting function is set up to minimize sensitivity at low frequencies whilst W_2 is designed to minimize the complementary sensitivity function at high frequencies.

The mixed optimization problem is the one that is typically used in H-infinity to ensure a robust design. This optimization problem can be solved using state-space techniques. One numerically stable algorithm for solving this problem is the Glover-Doyle algorithm. The reader is referred to good references on this subject such as *A Course in Linear H-Infinity Control Theory* or *Feeback Control Theory.* These works are listed in the references on Page 11.

The designs in this dissertation made use of a Matlab routine that implements a version of this Glover-Doyle algorithm (See Appendix 9.1.4 for a description of these routines). Direct applications of H-infinity theory were also applied in the cases of sensitivity minimization with no constraint on the complementary sensitivity function.

The theory presented here shall be revisited in the robot arm plant design chapter.

3.2 References

Maciejowski J.M. — Multivariable Feedback Control, Addison Wesley, 1989

Francis B.A. — A Course in Linear H-infinity Control Theory, Springer-Verlag, 1987

Doyle J.C., Francis B.A., Tannenbaum A.R. — Feedback Control Theory, Macmillan, 1992

4 The Robot Arm plant

4.1 Introduction

The robot arm is a non-linear second order plant. Physically it is not very complex, consisting of 1 motor, a shaft and a weight. A schematic of the plant is shown in Figure 4.1.

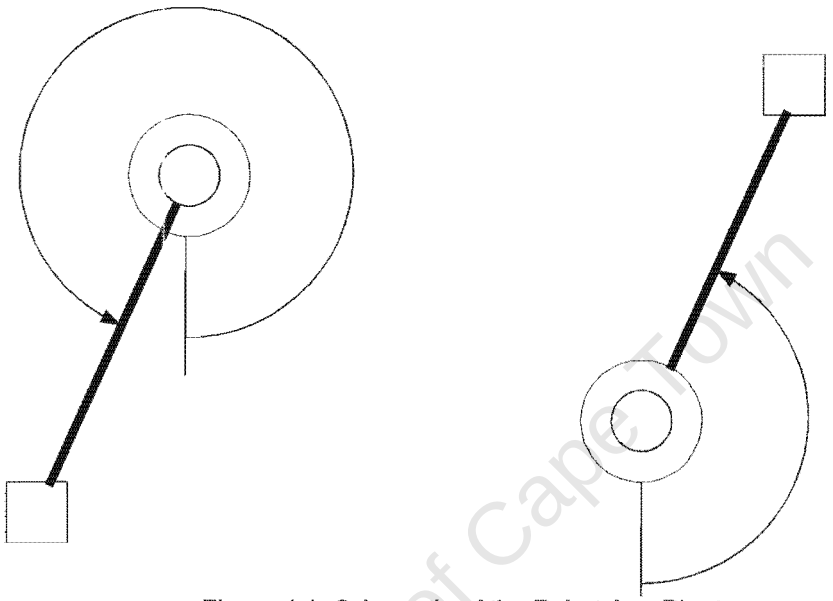


Figure 4.1 Schematic of the Robot Arm Plant

The aim is to control the position of the arm. This angle is measured counterclockwise from the start position, where the shaft is hanging vertically down. The plant is stable in the region 0 to 90 and 270 to 360 degrees and unstable in the 90 to 270 degree region. The controller that is designed should stabilize the plant in both regions. The calibration range is: 0 – 360 corresponds to 0 to 6.96 Volts.

The model for the plant was derived by performing step tests on the open loop plant in the stable region. The model for the unstable region can be derived from the stable model via a sign change on certain terms. Refer to appendix 9.2.1 for a discussion on the structure of the model derived for the plant.

Five unit step tests were performed on the open loop plant and the nominal model derived for the stable region was:

$$G(s) = \frac{0.4127}{1 + 0.0402 \cdot s + 0.0138 \cdot s^2} \frac{V}{V}$$

The corresponding model for the unstable region is predicted by theory to be:

$$G(s) = \frac{0.4127}{1 + 0.0402 \cdot s + 0.0138 \cdot s^2} \frac{V}{V}$$

The main limitation in the robot arm is the input range. The input to the plant is limited to –10 Volts to +10 volts. This factor must be taken into account in the design process, since a high gain controller will cause the input to exceed this limit very quickly. Refer to Appendix 9.2.1 for a model of the plant including uncertainties on the parameters.

4.2 H-infinity design

4.2.1 Introduction

The constraints for the robot arm design involved minimizing the sensitivity and complementary sensitivity functions in order to achieve a bounded input response.

These minimizations can be expressed as follows:

$$\|W_1 \cdot S\|_\infty < 1$$
$$\therefore \|S\|_\infty < \|W_1^{-1}\|_\infty$$
and
$$\|W_3 \cdot T\|_\infty < 1$$
$$\therefore \|T\|_\infty < \|W_3^{-1}\|_\infty$$

where W_1 is the weighting function on the sensitivity function and W_3 is the weighting function on the complementary sensitivity function.

Graphically these expressions imply that the sensitivity function of the closed loop system should lie below the inverse of the W_1 weighting function and likewise for the complementary sensitivity function.

4.2.2 *Weighting functions*

The weighting functions are based on the general formula for a second order filter.

The expression for a general second order filter is as follows:

$$W = \frac{\beta \cdot \left[\alpha \cdot s^2 + 2 \cdot z_1 \cdot w_{1c}\sqrt{\alpha} \cdot s + w_{1c}{}^2\right]}{\beta \cdot s^2 + 2 \cdot z_2 \cdot w_{1c}\sqrt{\beta} \cdot s + w_{1c}}$$

where

β = DC Gain (Controls disturbance rejection)

α = High Frequency Gain (controls peak overshoot)

w_{1c} = Cross-over frequency (-3 dB point)

z_1, z_2 = Damping ratios at corner frequencies

This filter will be expressed using the following format:

GenFilter(DC Gain, High Frequency Gain, Cross Over Frequency, Damping Ratio 1, Damping Ratio 2)

4.2.3 Robot Arm Design 1

Design 1 uses the following weighting functions:

$$W_1^{-1} = GenFilter(500,1,10,0.7,0.7)$$
$$W_3^{-1} = GenFilter(1,10,100,2.8,1.3)$$

The plots of these weighting functions are shown in Figure 4.2 below. A number of different values were chosen for the weighting functions. Based on the general shape of the output for a workeable solution, the curves were adjusted to minimize the sensitivity and complementary sensitivity functions to a greater degree. The result were the values above.

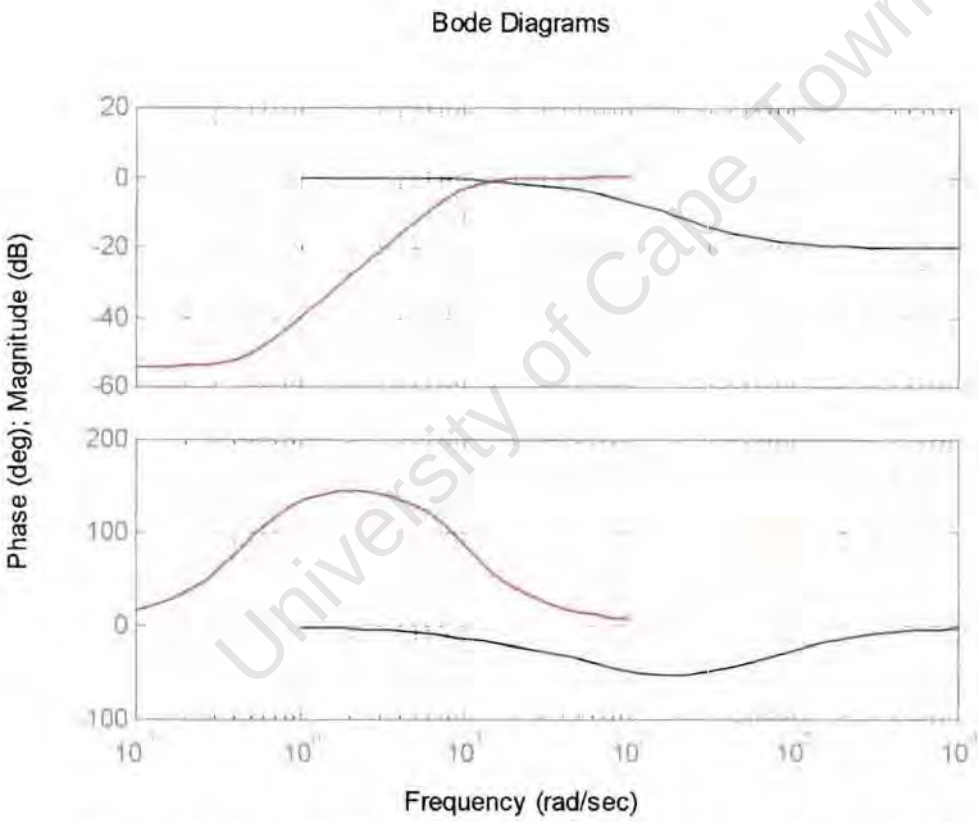


Figure 4.2: Plot of W_1^{-1} (Red) and W_3^{-1} (Blue) for Design 1 of the Robot Arm

The plot of Figure 4.2 indicates that the sensitivity function is minimized up to 1 rad/s, whilst the complementary sensitivity function is minimized for frequencies above 100 rad/s.

The sensitivity minimization in Figure 4.2 will allow for setpoint tracking at low frequency, whilst the complementary sensitivity minimization ensures good disturbance rejection for high frequency interference such as noise.

The weighting functions are thus ready for implementation in an H-infinity design routine. Unfortunately, the base weighting functions specified above do not result in a valid H-infinity controller. For this reason the weighting functions were shifted in order to obtain a valid H-infinity controller. The multiplier was increased slightly from 1 until the controller was valid. As the multiplier is increased, the weighting functions shift upwards. This implies that the sensitivity minimization becomes less stringent. For this reason, the multiplier should be kept small to make the weighting function perform as closely to the intended design as possible.

$$\text{If } W_1^{-1} = \frac{Sn0}{Sd0} \text{ and } W_3^{-1} = \frac{Tn0}{Td0}$$

then the shifted weighting functions are

$$1.7 \cdot \frac{Tn0}{Td0} \text{ and } 2 \cdot \frac{Sn0}{Sd0}$$

These weighting functions resulted in a controller that produced a closed loop step response as shown in Figure 4.3. Although the output response is satisfactory, the input response is exceedingly high. This is shown in Figure 4.4. The magnitude of the input response is only satisfactory 2 seconds after the step has taken place. If the input is clipped to a maximum of 10 volts, then the closed loop system becomes unstable. Hence this base design requires some modification before it can be used.

The input response shown contains a large amount of oscillation and aliasing. The reason for the introduction of these oscillations is the modification to the plant to satisfy the criteria of the Glover-Doyle algorithm for H-infinity design. This criterion states that the D term in the state-space representation of the plant should not be zero. Thus in order to design for a plant of this kind, the D term must be given a value. A typical way to determine this value is by calculating the modulus of the plant transfer function at a particular frequency. For example, if the control is to take place below 100 rad/s, then the D term can be set to the modulus of the plant frequency transfer function at 100 rad/s (see appendix 9.1.4).

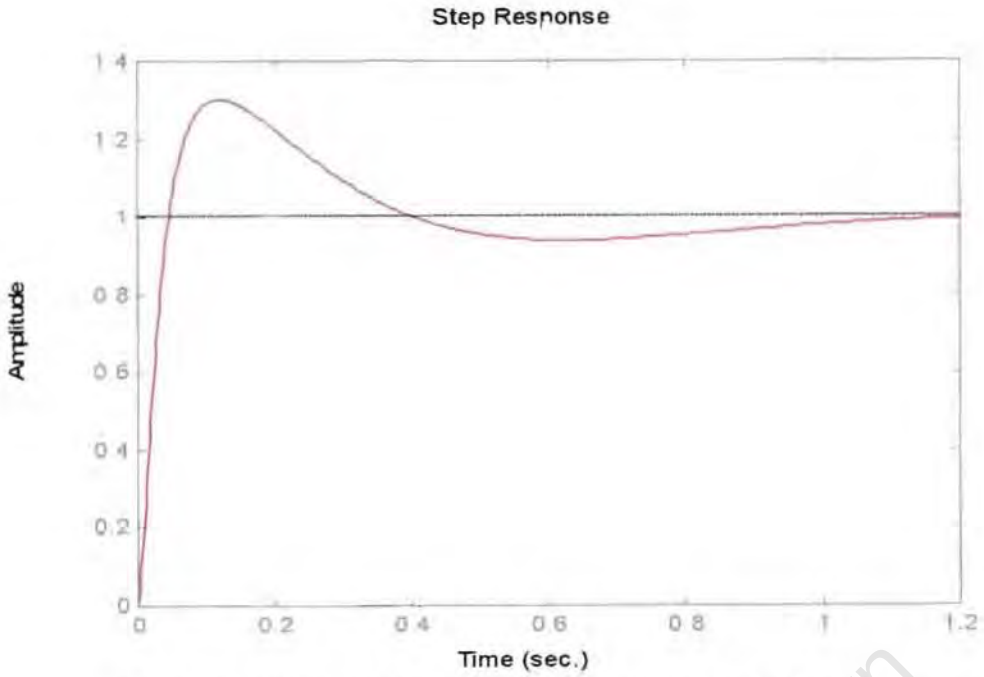


Figure 4.3: Closed loop step response for design 1 of the robot arm

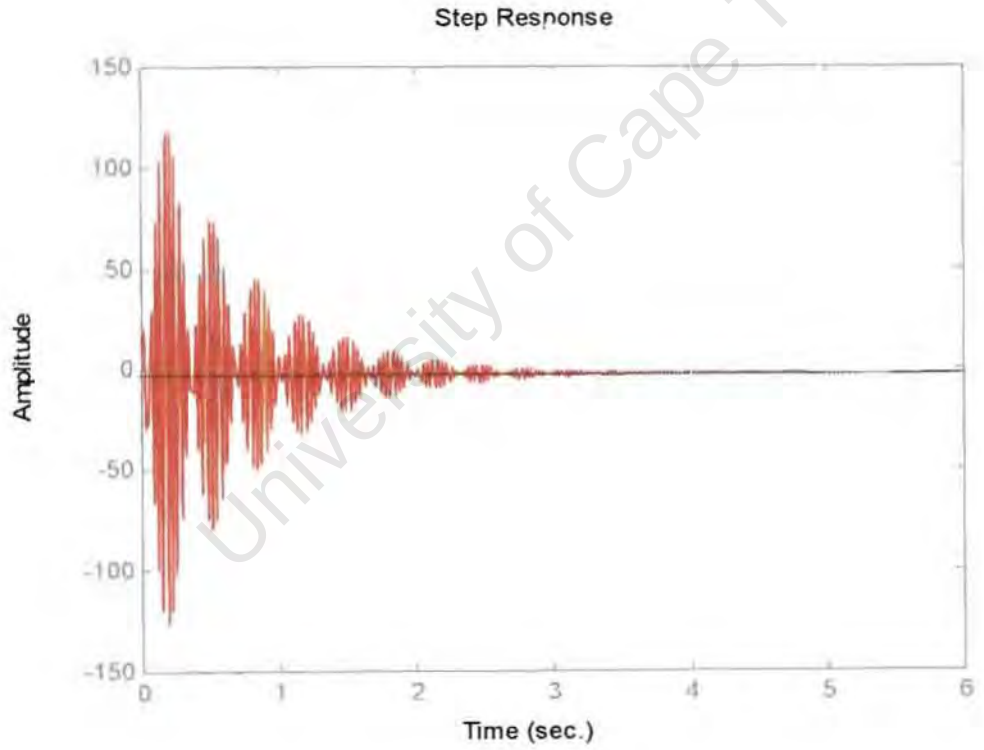


Figure 4.4: Input response for design 1 of the robot arm

4.2.4 Robot Arm Design 2

Design 2 makes use of the following weighting functions:

$$W_1^{-1} = GenFilter(500,1,10,0.7,0.7) = \frac{Sn0}{Sd0}$$

$$W_3^{-1} = 1 - \frac{Sn0}{Sd0} = \frac{Tn0}{Td0} \text{ with } \mathrm{Tn}(1) = 0.1$$

A design was effected using

$$5 \cdot \frac{Sn0}{Sd0} \text{ and } 3.5 \cdot \frac{Tn0}{Td0}$$

The plant was slightly preprocessed to account for a 50ms sampling time. This preprocessing also solves some criterion issues required for the Matlab H-infinity routine. The plots shown involve this modified plant controller. In the digital controller section, the original plant is combined with a digital controller. This configuration forms the basis for physical implementation of the controller.

Bode plots of the desired sensitivity and complementary sensitivity functions are shown in figure 4.5.

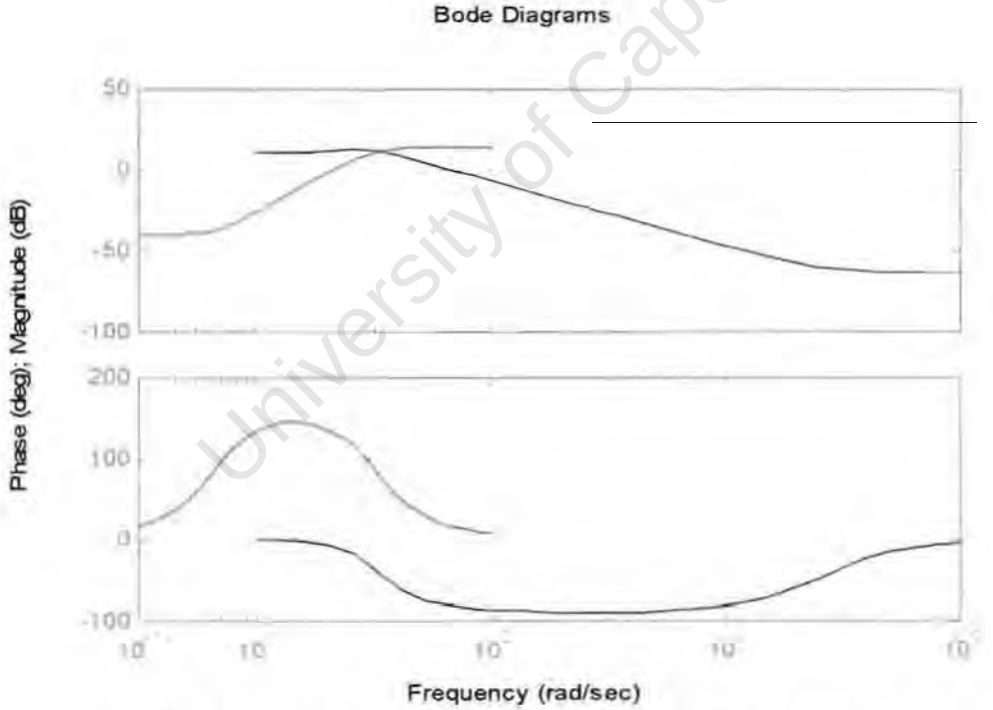


Figure 4.5: Bode plots of the 5*Sn0/Sd0 (Red) and 3.5*Tn0/Td0 (Blue)

The closed loop step response and input response for the controlled plant is shown in Figure 4.6 and 4.7 respectively. The corresponding bode plots for these responses are shown in Figures 4.8 and 4.9 respectively (together with the desired bode responses).

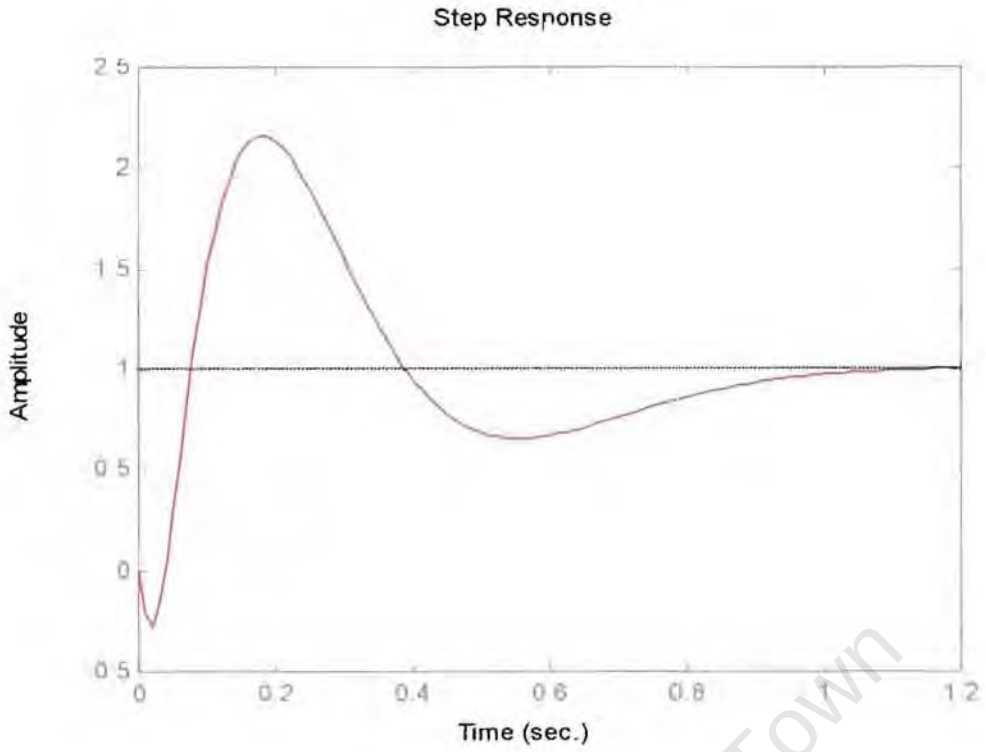


Figure 4.6: The closed loop step response for design 2 of the Robot Arm

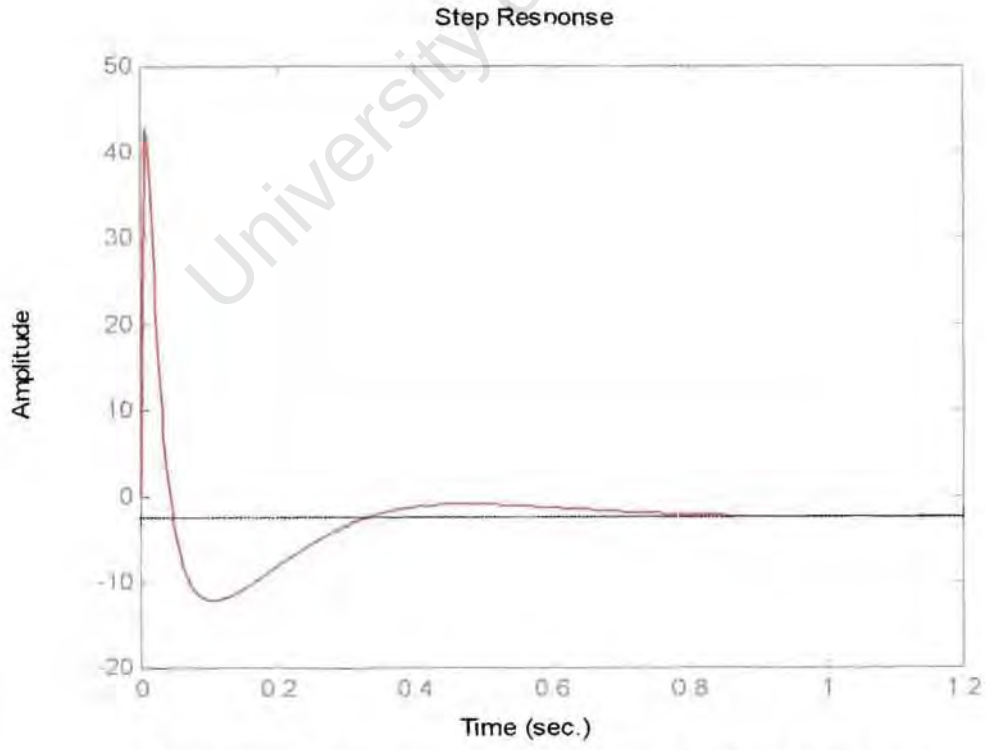


Figure 4.7: The input response for design 2 of the Robot Arm

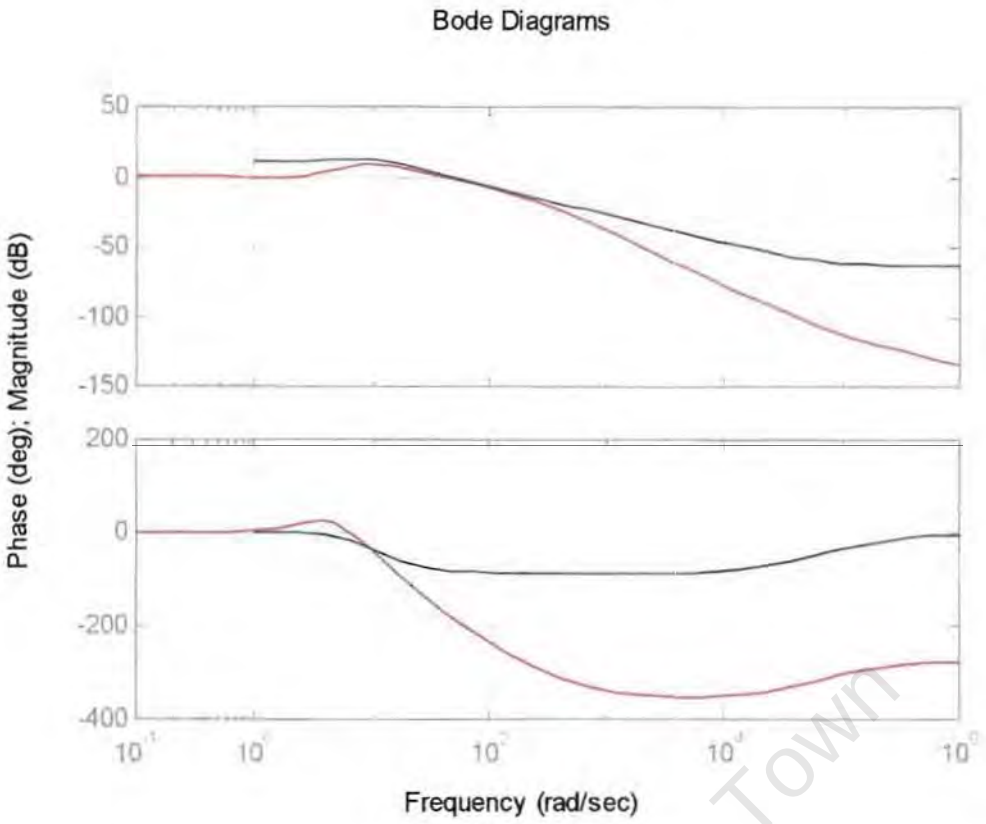


Figure 4.8: Bode plot of the complementary sensitivity function (Red) together with the desired complementary sensitivity function (Blue) for design 2 of the Robot Arm

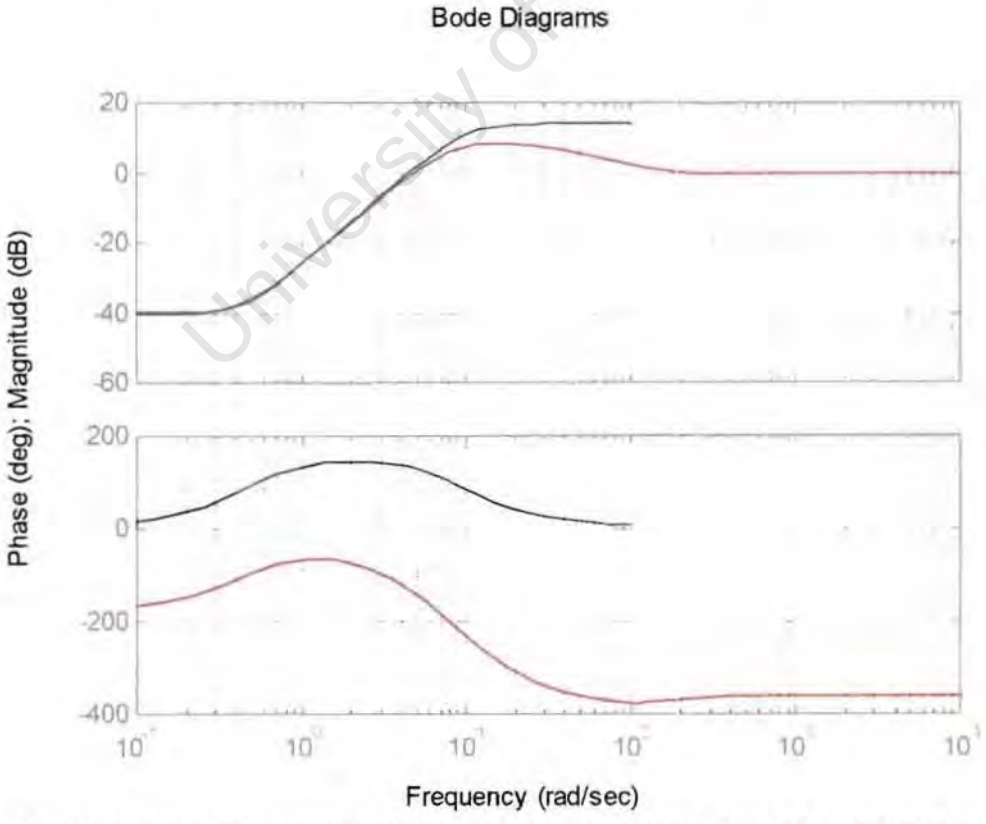


Figure 4.9: Bode plot of the sensitivity function (Red) and the desired sensitivity function (Blue) for design 2 of the Robot Arm

4.3 Analysis of Robot Arm H-infinity Designs

Of the two designs presented above, design 2 was chosen as the controller to use for the plant. This is the design that shall be used for the remainder of the dissertation. In the next section a digital implementation of this controller shall be investigated. The following observations can be made about the design from the plots presented in the previous section:

- The closed loop step response has a large amount of overshoot. However, this is a compromise in order to keep the input within the –10 to 10 volt limits

- The closed loop system has a settling time of approximately 1.2 seconds. The stable open loop plant settles within 4 seconds. Thus the controlled system is 75% faster than the open loop plant and this is thus satisfactory

- The desired sensitivity function requires a sensitivity less than -30 dB up to 1 rad/s. This corresponds to a maximum steady state error of approximately 0.0316 for a unit step. The actual closed loop response has a steady state error of less than 0.01 that corresponds to the –40 dB section of the sensitivity plot.

- The desired complementary sensitivity function has a gain of approximately 1 up to 3 rad/s. This indicates that the input is equal to the output up to this frequency. The closed loop response forces robust performance in the face of disturbances. Uncertainty increases with frequency, as noise is a high frequency disturbance. The complementary sensitivity function compensates for this increased uncertainty by penalizing higher frequencies more than lower frequencies.

4.3.1 *Gain and Phase margin analysis*

A bode plot indicating the gain and phase margins for design 2 of the robot arm is shown in Figure 4.10.

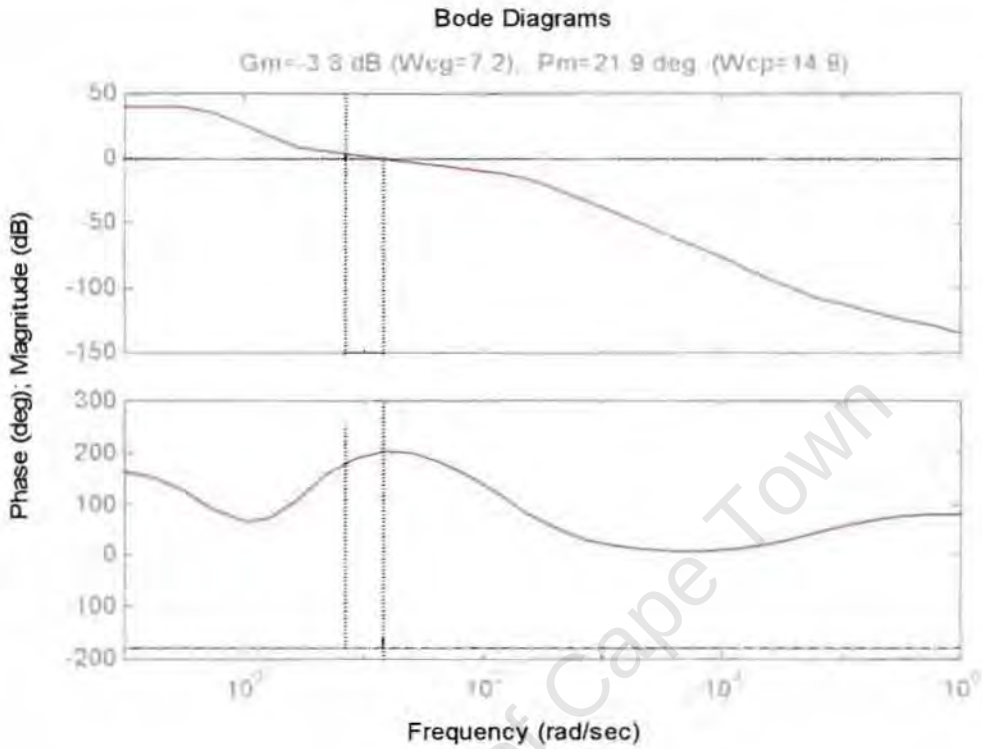


Figure 4.10: Gain and Phase Margin of design 2 for the Robot Arm

From Figure 4.10, the gain margin is –3.8 dB and the phase margin is 21.9 degrees. This phase margin is large enough to ensure stability in the face of phase perturbations: a phase perturbation greater than 21.9 degrees will be required to destabilize the closed loop.

The gain margin of –3.8 dB (actual gain of 0.64565) provides a large margin for gain perturbations: a change in gain of –9.01 dB (actual gain of 0.3543) is required to destabilize the closed loop system. This corresponds to a 35.43% change in gain.

4.3.2 Nyquist analysis

Nyquist plots of the open loop system including the controller are shown in Figure 4.11 and Figure 4.12.

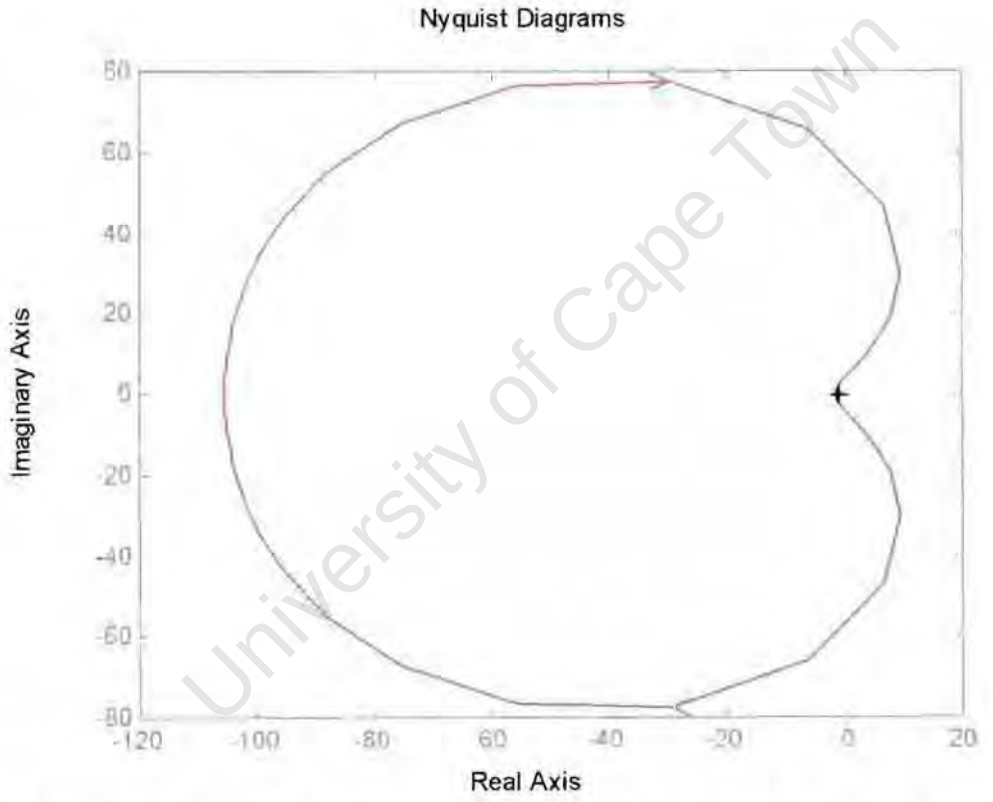


Figure 4.11: Nyquist plot of the controlled open loop system for design 2 of the Robot Arm

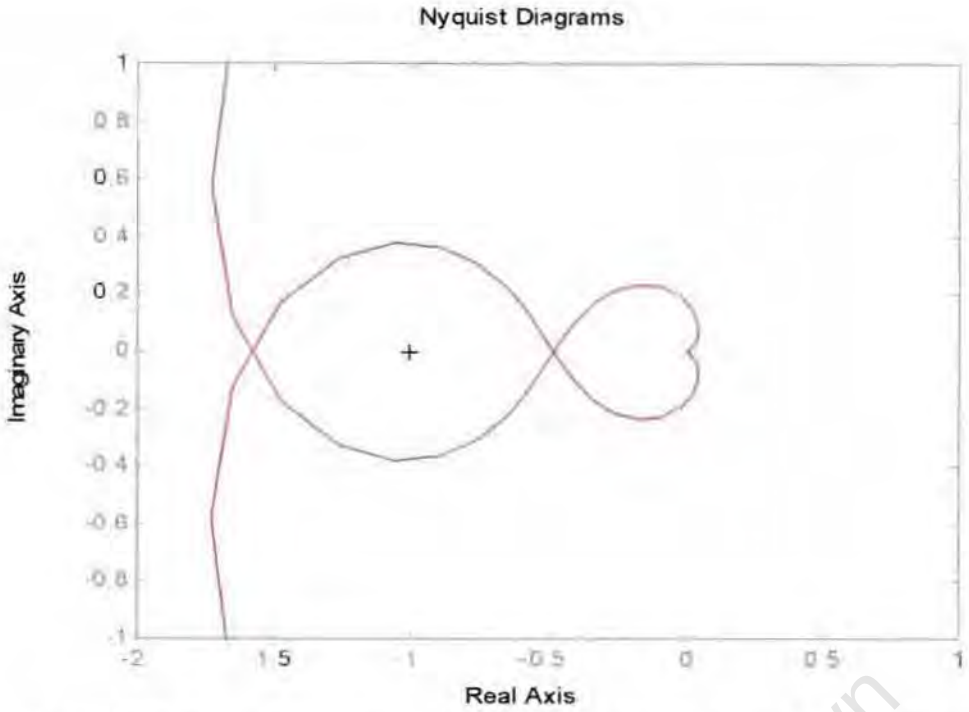


Figure 4.12: Zoomed in section of the Nyquist plot for design 2 of the Robot Arm

The nyquist plots confirm that the closed loop system will be stable. The number of encirclements of the critical point (-1,0) is equal to one, which corresponds to the number of unstable open loop poles.

4.4 Digital Controller Design for the Robot Arm

It is intended to implement the controller for the Robot Arm plant using a computer. Thus the plant will be sampled. The designs presented in the previous sections were done in continuous time. Using the continuous design as a basis, the system will be investigated in discrete time.

A block diagram for the sampled system is shown in Figure 4.13.

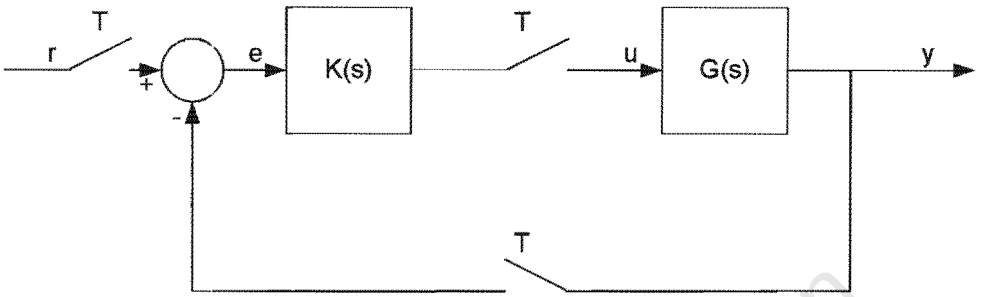


Figure 4.13 Digital Control Loop

The initial choice for the sampling time is 50ms. As the plant has oscillatory poles with T=0.75 s, this sampling time is adequate. There will be approximately 10 samples per pole. However, the main issue to consider is whether system will be stable in closed loop.

This system is investigated through digital simulation using Matlab. The steps involved to simulate this system using the continuous time case as a basis are as follows:

1. Convert the plant to discrete time using a Zero Order Hold circuit. This Zero Order Hold circuit is contained in the digital to analog converter in the computer

2. Convert the controller to discrete time using the Tustin bilinear transform

If the system is simulated digitally then it is found to be unstable. The main reason for this instability is the breaking of the loop between K and G. When the loop is broken in this way, a delay is introduced into the system. In discrete time this corresponds to placing a pole at zero in the z-plane.

If the sampling time is reduced to 20ms, then the closed loop system is stable in simulation. A 30ms sample time will also stabilize the system in the unstable region, but not in the stable region. Hence the chosen sampling time is 20ms. The routine that implements the digital controller on the computer can complete its loop within 20ms on a 150Mhz Intel Pentium Pro processor.

The input and output responses for different step sizes in the stable and unstable regions are shown in Figures 4.14 to 4.17.

Figure 4.14 and 4.15 show input and output responses for steps of size 1, 2, 3 and 4 Volts. Figure 4.16 and 4.17 show input and output responses for steps of size 1, 1.5 and 1.7 Volts.

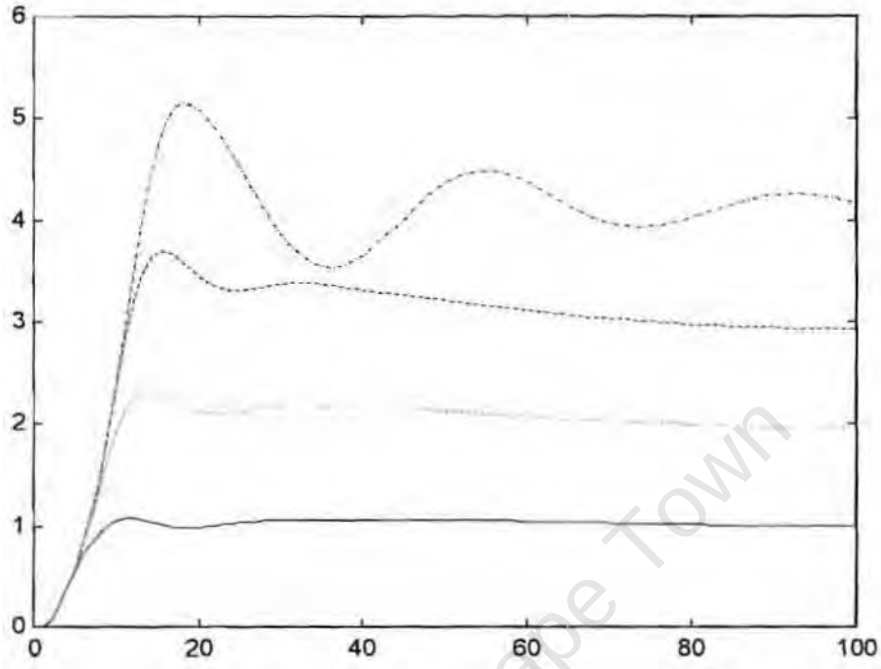


Figure 4.14: Output step responses for the digitally controlled Robot Arm in the stable region

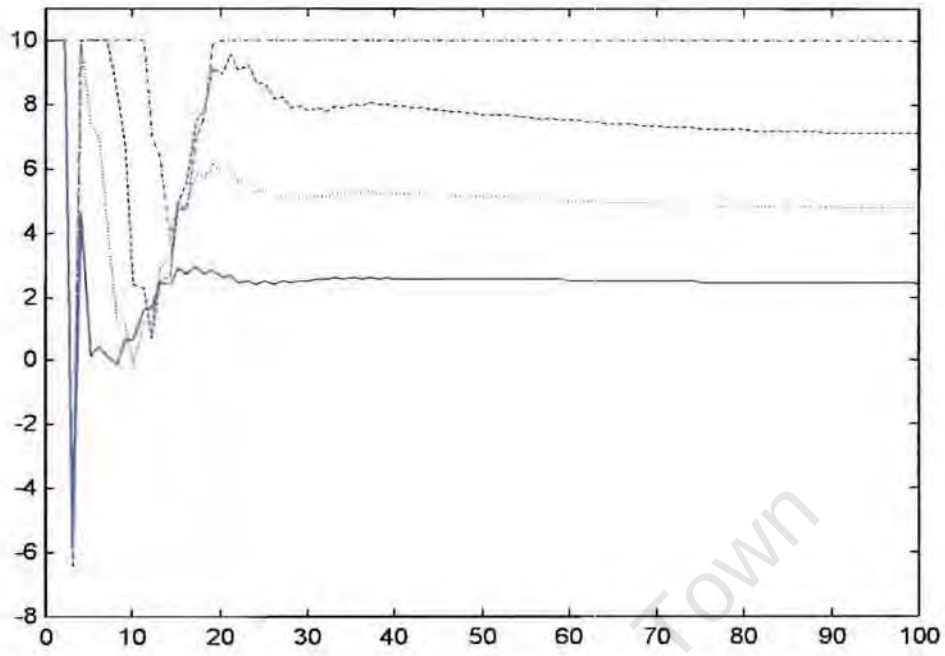


Figure 4.15: Input step responses for the digitally controlled Robot Arm in the stable region

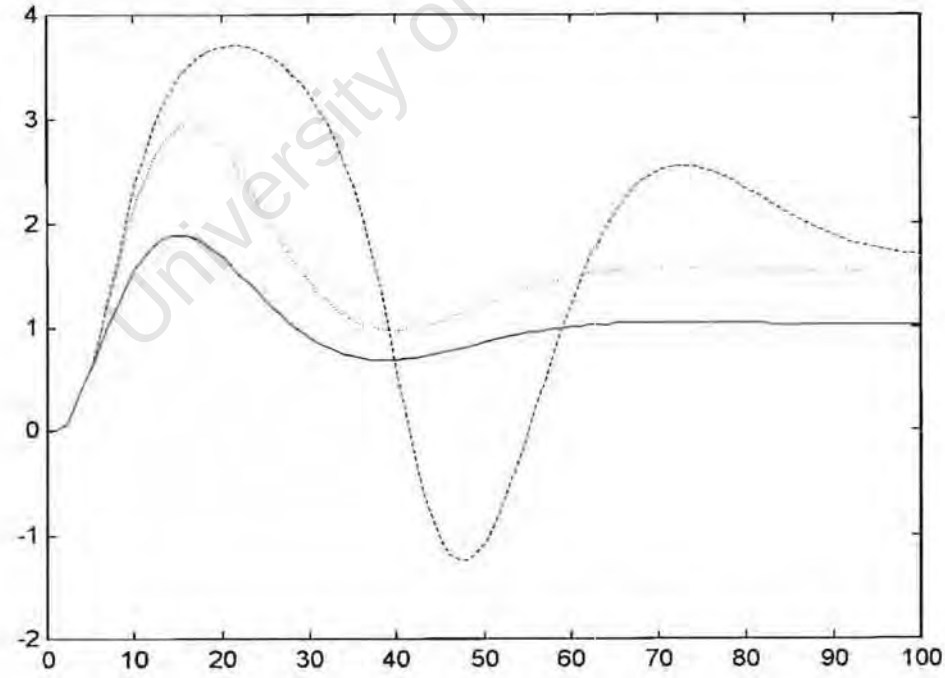


Figure 4.16: Output step responses for the digitally controlled Robot Arm in the unstable region

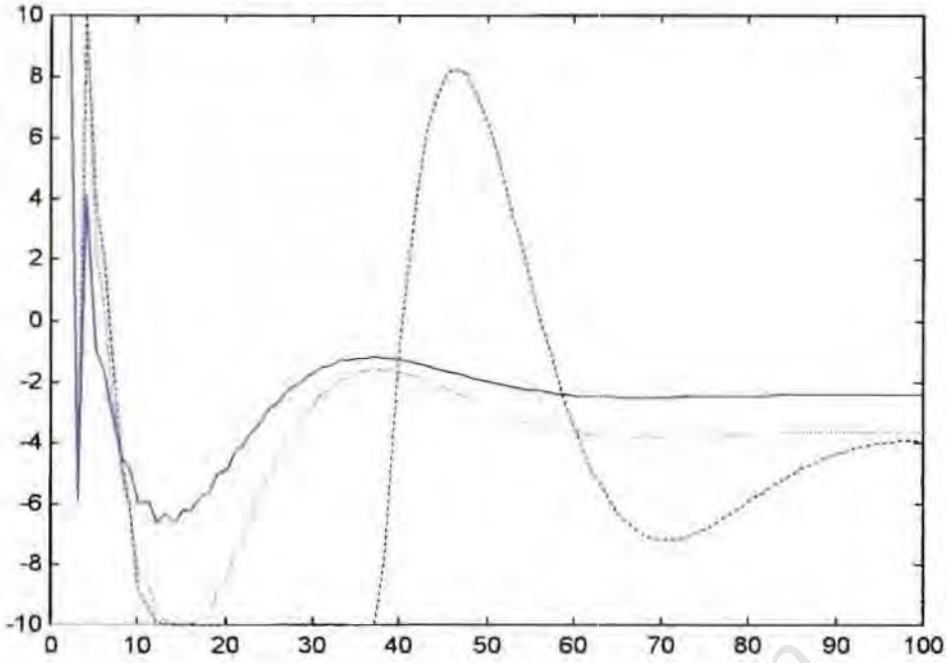


Figure 4.17: Input step responses for the digitally controlled Robot Arm in the unstable region

In all the above cases, the input was clipped at –10 to 10 Volts. From the plots it is clear that the stable region is well controlled with this controller. It is only when the input step reaches 4 volts that the input hits the 10 volt rail for the majority of the time.

In the open loop unstable region, however, the small input limit affects the output response quite dramatically. The steps have to be limited to 1.7 volts. If the step becomes larger, then the system simply goes unstable as 10 volts is insufficient to bring the system to stability with such a large step input. Furthermore, the output step responses deteriorate rapidly as the step setpoint gets larger.

The limit on the input energy is therefore quite a prominent problem in the robot arm plant. However, in comparison to the unstable open loop plant, this design can still be treated as satisfactory.

4.5 Robot Arm Plant Models

This section is a summary of the transfer functions for the Robot Arm plant including the transfer function for the controller generated in design 2.

Plant Transfer Functions (Nominal Models)

$$G(s) = \frac{0.4127}{1 + 0.0402 \cdot s + 0.0138 \cdot s^2} \text{ and}$$

$$G(s) = \frac{-0.4127}{1 - 0.0402 \cdot s - 0.0138 \cdot s^2}$$

Controller Transfer Function (design 2 of the Robot Arm Plant)

$$C(s) = \frac{q_6^0 \cdot s^6 + q_5^0 \cdot s^5 + q_4^0 \cdot s^4 + q_3^0 \cdot s^3 + q_2^0 \cdot s^2 + q_1^0 \cdot s^1 + q_0^0 \cdot s^0}{p_6^0 \cdot s^6 + p_5^0 \cdot s^5 + p_4^0 \cdot s^4 + p_3^0 \cdot s^3 + p_2^0 \cdot s^2 + p_1^0 \cdot s^1 + p_0^0 \cdot s^0}$$

where

Numerator		Denominator	
Q_6^0	-0.0000448675	P_6^0	1.0000000000
Q_5^0	453.7661307045	P_5^0	2022.5859937073
Q_4^0	30543021.9355337000	P_4^0	641140.0816065320
Q_3^0	642725722.2467380000	P_3^0	61667185.4113773000
Q_2^0	4491215385.0456200000	P_2^0	440026902.6730430000
Q_1^0	13168465210.1525000000	P_1^0	26365724 3.1396800000
Q_0^0	20552318849.3999000000	P_0^0	80308161.8533135000

Controller Transfer Function (discrete time version with sampling time of 20ms)

$$C(z) = \frac{q_6^0 \cdot z^6 + q_5^0 \cdot z^5 + q_4^0 \cdot z^4 + q_3^0 \cdot z^3 + q_2^0 \cdot z^2 + q_1^0 \cdot z^1 + q_0^0 \cdot z^0}{p_6^0 \cdot z^6 + p_5^0 \cdot z^5 + p_4^0 \cdot z^4 + p_3^0 \cdot z^3 + p_2^0 \cdot z^2 + p_1^0 \cdot z^1 + p_0^0 \cdot z^0}$$

where

Numerator		Denominator	
Q_6^0	24.75	P_6^0	1.00
Q_5^0	-39.83	P_5^0	-1.362
Q_4^0	33.00	P_4^0	-0.9064
Q_3^0	79.49	P_3^0	1.445
Q_2^0	-7.924	P_2^0	0.1996
Q_1^0	-39.66	P_1^0	-0.2947
Q_0^0	16.18	P_0^0	-0.08192

4.6 Digital Implementation of Controller for Robot Arm Plant

4.6.1 Introduction

A number of factors will influence the performance of the designed controllers on the physical plant. These factors are listed below:

- The real plant is non-linear. Hence the model of the plant changes as the shaft position rotates through 360 degrees. The controllers were designed for a linearised plant centred at zero and 180 degrees for the stable and unstable regions respectively.
- The continuous time model derived for the plant was obtained using an ADC/DAC combination. For this reason the zero order hold circuit contained in the ADC/DAC unit will affect the constants derived in the plant model. In converting the controller to a discrete time controller, the plant should not actually be combined with a zero order hold circuit when performing discrete time simulation.
- Many controllers can be designed to stabilize the plant in practice. However, implementing them using a computer imposes a physical restriction on the order of the controller. The higher the order of the controller, the longer the software simulation will take. If this calculation is greater than the sampling time, then the controller will not perform as intended. The reason for this is that the samples obtained from the physical plant will not necessarily be at the same rate as that at which the controller is being simulated. This difference in sampling times will result in aliasing and excessive oscillations in the physical plant.

When the designed discrete time controller of section 4.5 was used to drive the physical plant, it was found that the system was stabilized in the stable region, but not in the unstable region. However, when the continuous time controller was used on the physical plant, the results were more satisfactory, but still very oscillatory. The reason for this, is that the continuous time controller requires a sampling time of 1 millisecond as a result of the pole at –1653. The control loop however requires in excess of 10 milliseconds to complete. The result is that the plant ends up being sampled at 20 milliseconds while the controller is sampled at 1 millisecond. These issues are not a consequence of the controllers and do not reflect accurately on the real performance of the controller, since the physical implementation does not allow for a fast enough sample time.

4.6.2 *Physical controller solution*

To solve these problems it was necessary to generate a new discrete time controller derived from the original continuous time controller.

The simulation of a continuous time system in software can be achieved using a Runge Kutta algorithm. This is very expensive in processor time and cannot be completed in suitable duration. The simulation of a discrete time controller, however, can be achieved by simple linear algebra.

The solution to the physical implementation therefore lies in obtaining a discrete time controller that mimics the continuous time controller as closely as possible. This can be achieved by imposing a very small sample time. Once again, this sample time cannot be too fast else the loop cannot be completed fast enough on an average PC. The sample time chosen was therefore the maximum sample time that would capture the –1653 pole of the controller, namely 1 millisecond.

The transfer function of this controller is shown below:

$$C(z) = \frac{q_6^0 \cdot z^6 + q_5^0 \cdot z^5 + q_4^0 \cdot z^4 + q_3^0 \cdot z^3 + q_2^0 \cdot z^2 + q_1^0 \cdot z^1 + q_0^0 \cdot z^0}{p_6^0 \cdot z^6 + p_5^0 \cdot z^5 + p_4^0 \cdot z^4 + p_3^0 \cdot z^3 + p_2^0 \cdot z^2 + p_1^0 \cdot z^1 + p_0^0 \cdot z^0}$$

where

Numerator		Denominator	
Q_6^0	3.644823913	P_6^0	1
Q_5^0	-7.423533978	P_5^0	-4.756399191
Q_4^0	-3.094237079	P_4^0	9.118937422
Q_3^0	14.01486777	P_3^0	-8.883057048
Q_2^0	-3.914123767	P_2^0	4.499740736
Q_1^0	-6.590674972	P_1^0	-1.04406394
Q_0^0	3.362878124	P_0^0	0.064842021

In software, using the above discrete time controller, the control loop can be completed in 1 millisecond using a 150 Mhz Intel Pentium Pro processor. The results of applying this controller to the physical plant are shown in the next section.

4.6.3 *Output plots for the controlled real plant*

A number of plots from the physical system will be shown. The corresponding expected simulation results will also be depicted. A discussion of the output waveform will follow all the plots. This discussion will highlight the performance of the controller in different areas of the output plot.

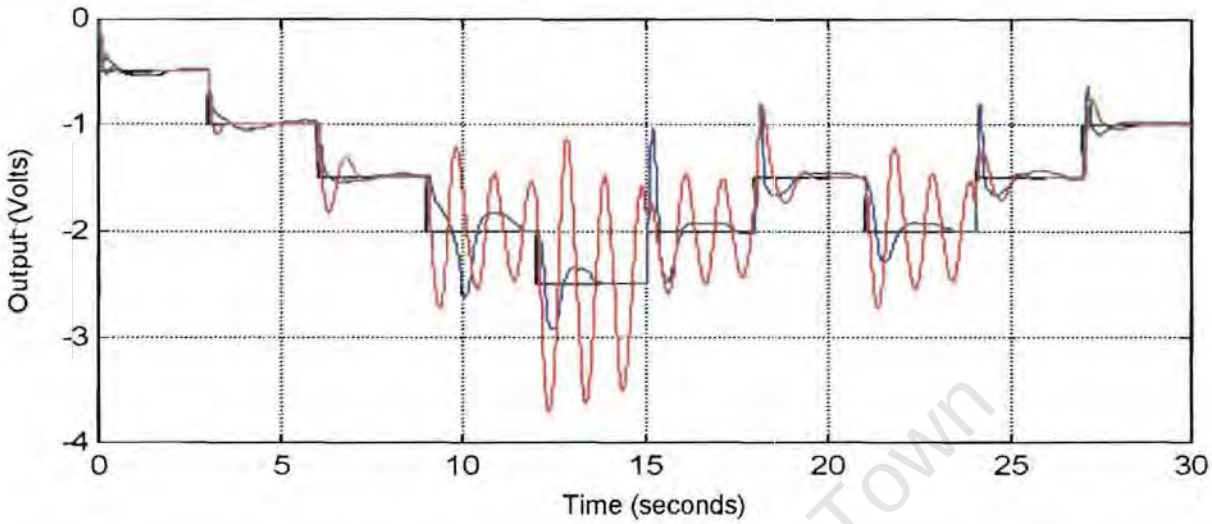


Figure 4.18 Output plots for physical plant showing the setpoint (black), the physical output (blue) and the non-linear simulation (red) results using a random waveform of step size -0.5 volts

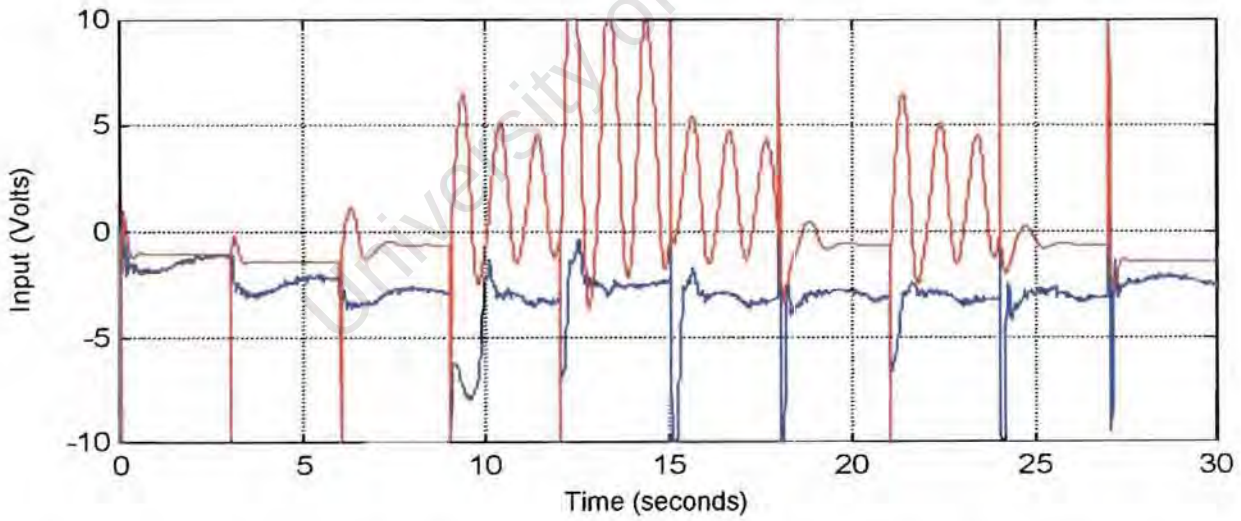


Figure 4.19 Plots of the input response for the setpoint of Figure 4.18. The simulated input response is in red and the physical input response is in blue.

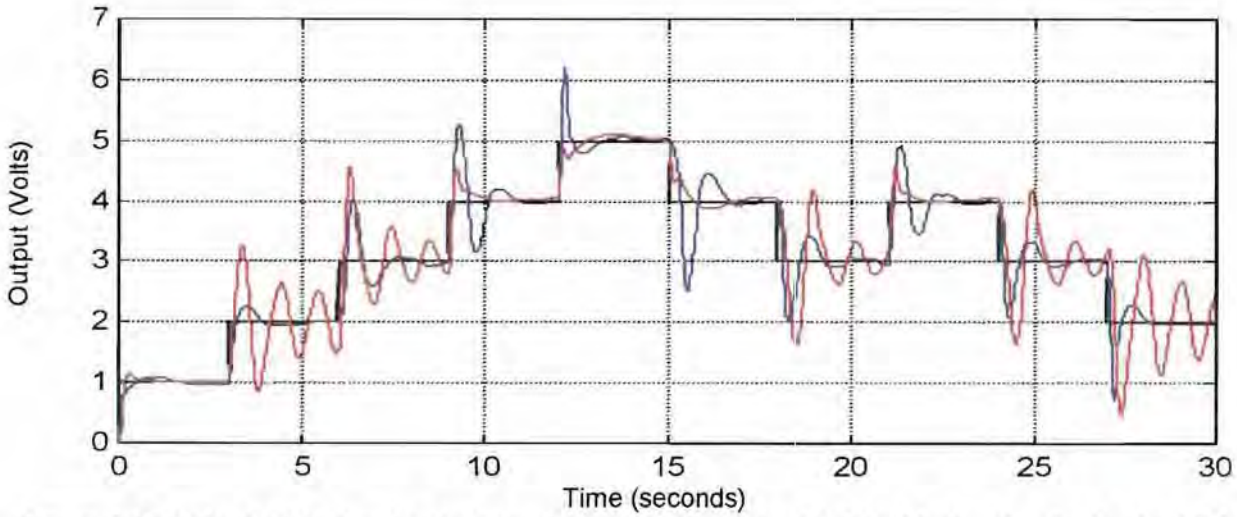


Figure 4.20 Output plots for physical plant showing the setpoint (black), the physical output (blue) and the non-linear simulation (red) results using a random waveform of step size 1 volt

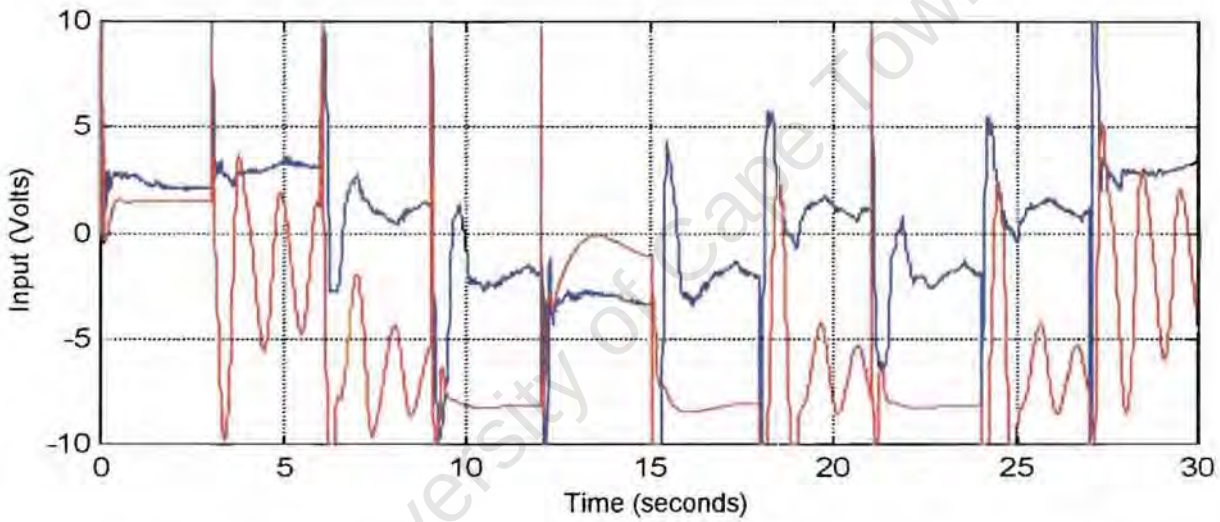


Figure 4.21 Plots of the input response for the setpoint of Figure 4.20. The simulated input response is in red and the physical input response is in blue.

Figures 4.18 to 4.21 illustrate the output results for the physical plant for a random step pattern that swings the arm clockwise and anticlockwise. From the output plots it is clear that the output peaks at each step transition. This is expected as a large input is applied to the plant due to the increased error between the setpoint and the output. The output then settles down towards the setpoint value. It is clear that the oscillations are more pronounced in different areas of the output plot. These plots also include the results predicted by the non-linear simulation of the plant using a discrete time controller on the continuous time plant.

The pronounced oscillations in certain sections of the output plots are due to the non-linearity of the robot arm plant.

This can be explained as follows:

The denominator of the robot arm model can be expressed as:

$\alpha + sB + s^2 J$

where $\alpha = MgL\cos(\theta)$ (refer to appendix 9.2.1)

For the purposes of modelling the system linearly, the dynamic parameter α is linearized at a particular angle. The angle chosen for this linearization is 0 degrees. The linearized model for the unstable region is then calculated at 180 degrees. The value of α is therefore equal to MgL and –MgL respectively. Since the controller is designed to compensate for oscillatory poles in the plant at a particular frequency, as the robot arm swings, the change in pole positions will cause the controller to either dampen or strengthen the oscillatory poles of the plant. The pole positions at 0 degrees and 180 degrees are shown below:

-1.4565+ j8.3870

-1.4565- j8.3870 at 0 degrees (the stable region)

and

-10.0928

7.1798 at 180 degrees (the unstable region)

The structure of these poles are characteristic of the stable and unstable regions. This implies that from 0 to 90 degrees and 270 to 360 degrees, the plant poles will always be stable and oscillatory. From 90 to 270 degrees the plant poles will always be on the real axis and consist of one stable and one unstable pole. Consider the root locus plots for the stable region in Figure 4.22 and the unstable region in Figure 4.23 for the linearised plant model.

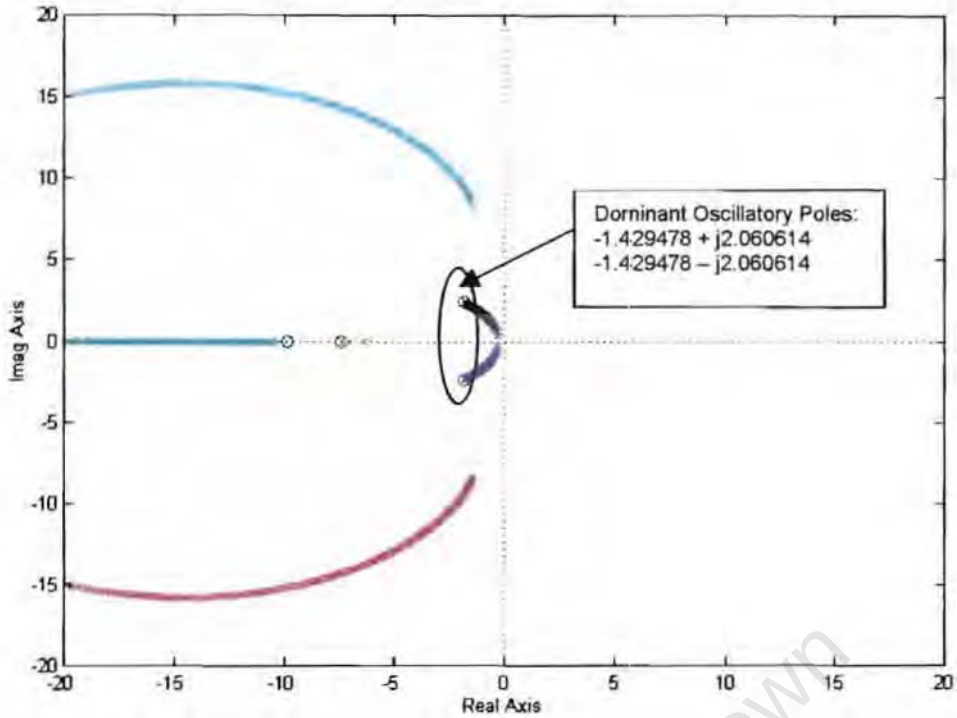


Figure 4.22 Root locus plots for the continuous time controller on the linearized plant model for the stable region of the robot arm

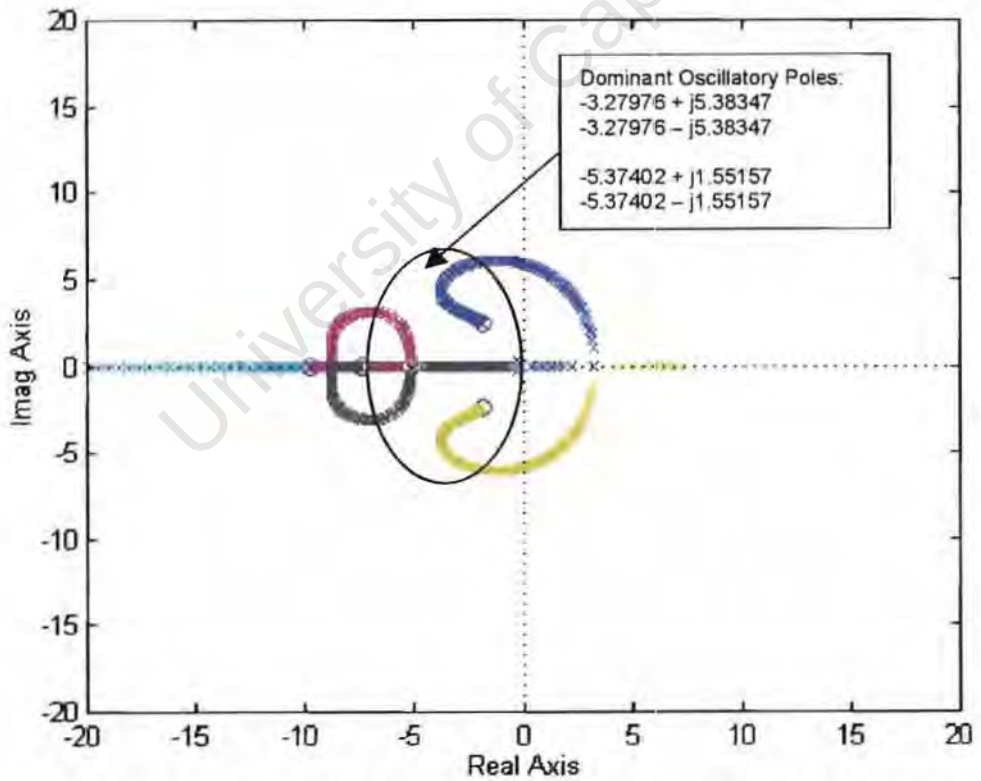


Figure 4.23 Root locus plots for the continuous time controller on the linearized plant model for the unstable region of the robot arm

From Figure 4.22 it is clear that the dominant oscillatory poles of the controlled system in the stable region are of low frequency: approximately 2 rad/s, with a settling time of approximately 4*(0.5) = 2 seconds.

From Figure 4.23 it is clear that the dominant oscillatory poles are of a higher frequency than in the stable case: approximately 5 rad/s, with a settling time ranging from 2 seconds to 10 seconds.

The high frequency and long settling time for the poles in the unstable region, accounts for the dominant oscillations in the different areas of the output plots of the controlled robot arm. The band from 1.7 volts to 5.26 volts and –1.7 volts to –5.26 volts forms the unstable region of the physical plant. Therefore, in this voltage range the oscillations are more pronounced as proven by the root locus plots of Figures 4.22 and 4.23

Despite these oscillations, the closed loop system does remain stable, and therefore the designed controller, although not optimal, does control the robot arm adequately.

The system was then perturbed using a square wave of size 2 volts. This will swing the robot arm between –90 and +90 degrees. The transition voltage is rather large at 4 volts and this setpoint therefore tests whether the controller does indeed stabilize the system when the robot arm is swinging at a fast pace.

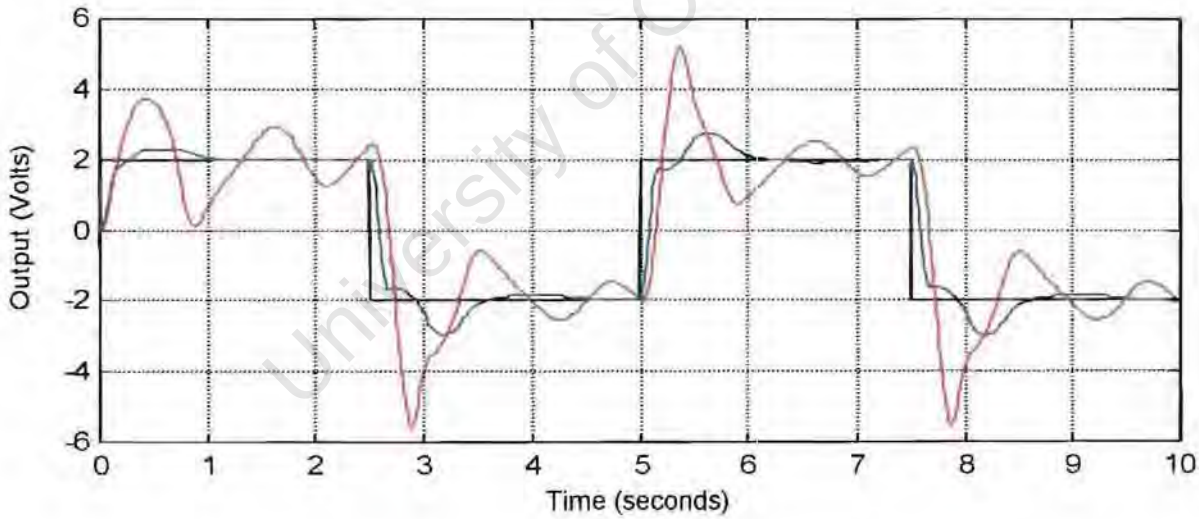


Figure 4.24 Output plots of the physical plant showing the setpoint (black), the physical output (blue) and the non-linear simulation (red) results for a square wave setpoint of size 2 volts

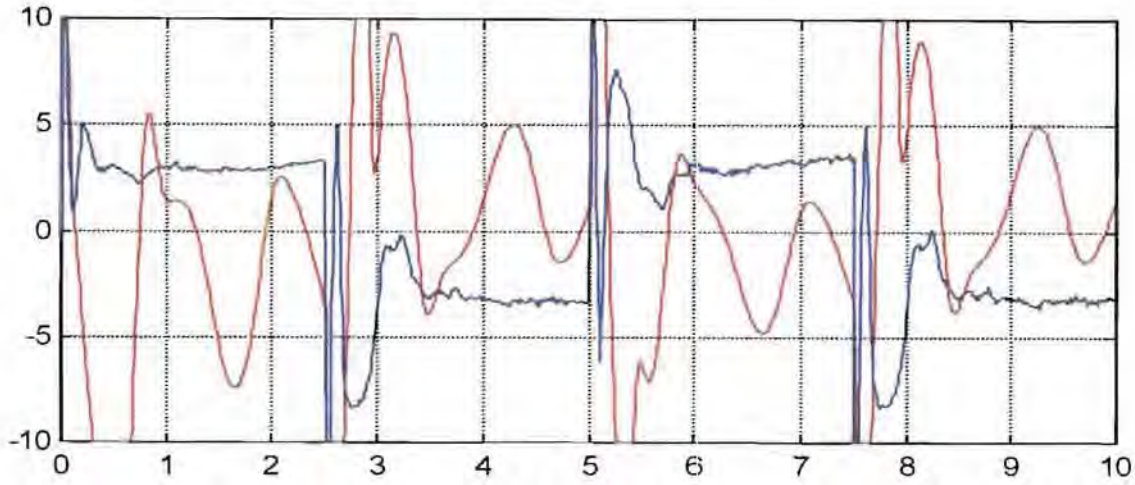


Figure 4.25 Plots of the input response for the setpoint of Figure 4.24. The simulated input response is in red and the physical input response is in blue.

From the plots shown in Figure 4.24 and Figure 4.25, it is clear the system is stabilized for the large 4 volts transition. The real input is also within limits and only peaks at the point of transition. The final test that was performed on the real system with this controller is a step test of 2 volts together with an external disturbance. The disturbance was generated by manually move the robot arm away from the setpoint whenever it tracked. The result was the system reacted and returned to the setpoint with an initial burst of oscillation. This result is shown in Figures 4.26 and 4.27.

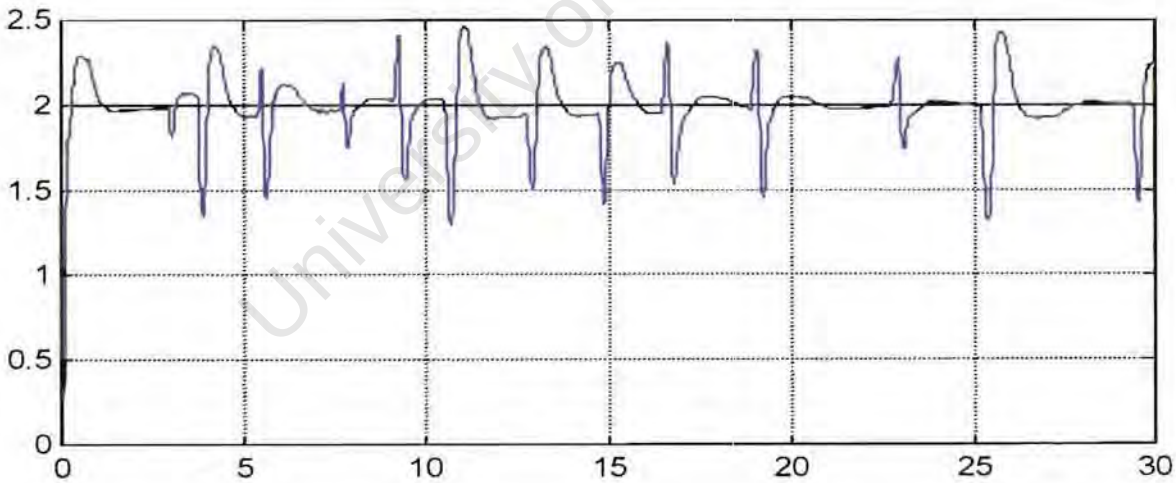


Figure 4.26 Output plots for the physical plant for a unit step of 2 volts together with an external disturbance.

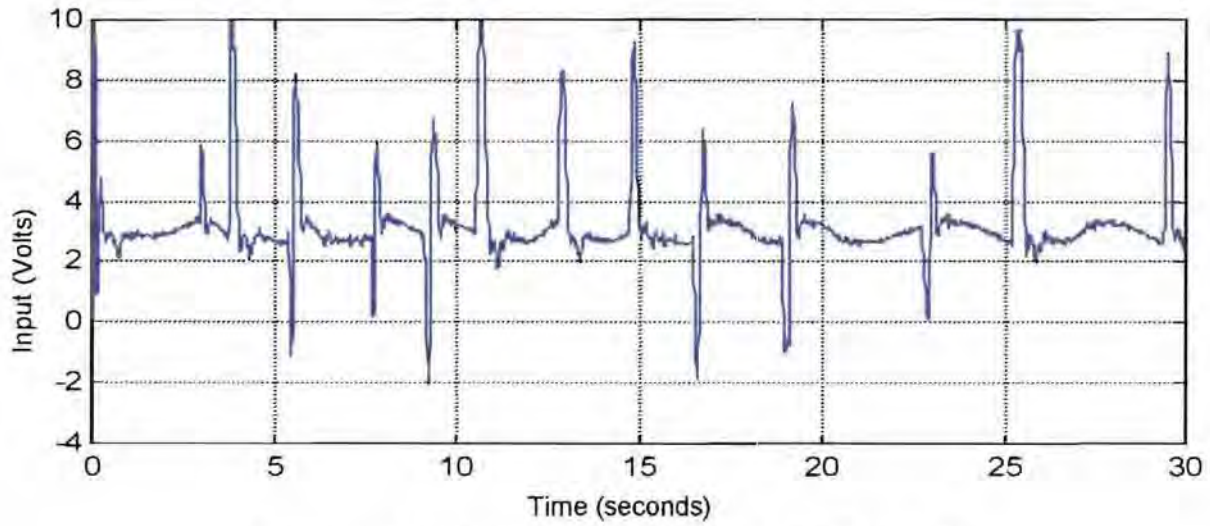


Figure 4.27 Input response for the setpoint of Figure 4.26

The final issue to test on the physical plant is the robustness of the controller for changes in controller coefficients. For this reason, a perturbation was applied to three coefficients in the controller. The magnitudes of these perturbations are: 200% on s^6 and 200% on s^5 in the numerator and 300% on s^6 in the denominator. The high power coefficients were changed as these have the greatest effect on the performance of the controller. The issue of this perturbing on the controller is further investigated in section 6.1.4. Once again the input used was a random waveform of step size 1 volt as in Figure 4.20.

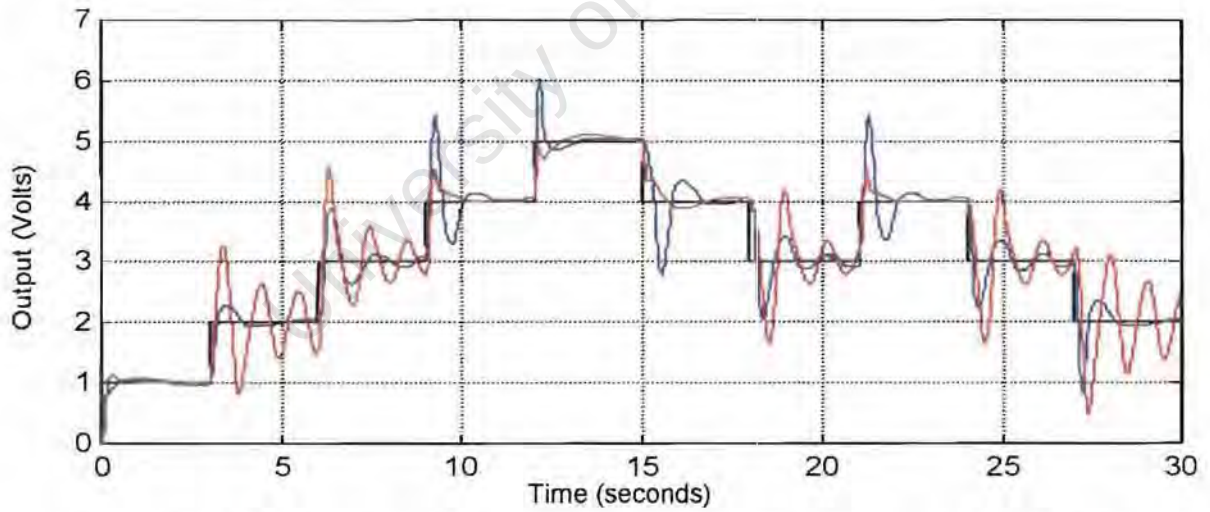


Figure 4.28 Output plots of the physical plant showing the setpoint (black), the physical output (blue) and the non-linear simulation (red) results for a random setpoint of step size 1 volt, using a perturbed controller

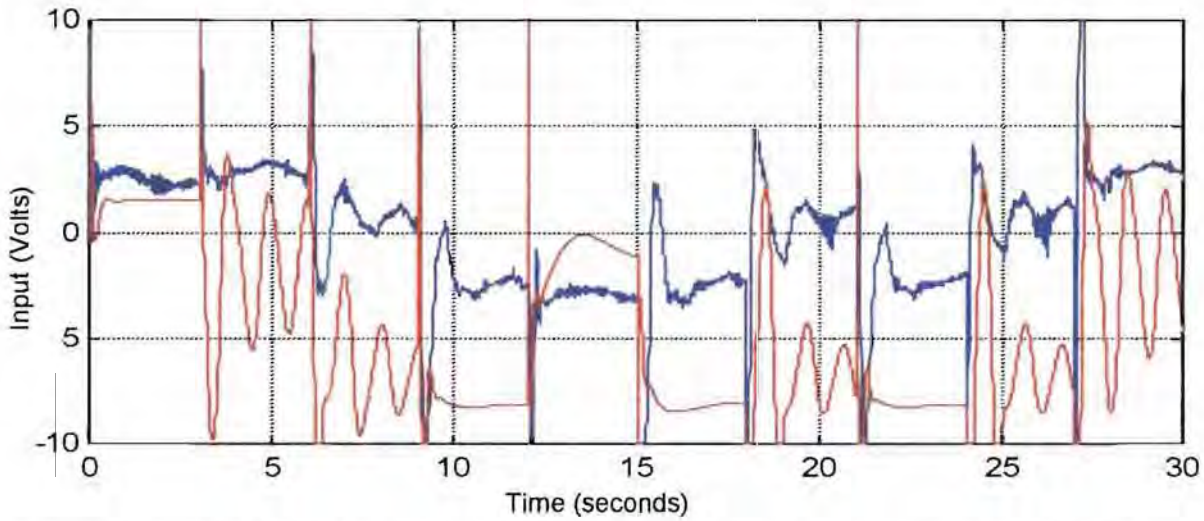


Figure 4.29 Plots of the input response for the setpoint and controller of Figure 4.28. The simulated input response is in red and the physical input response is in blue.

By comparing Figure 4.28 to Figure 4.20, there is no noticeable difference. The controller is therefore robust to rather large perturbations in the coefficients. This topic is revisited in section 6.1.4 when the parametric stability margin for this perturbed controller is discussed.

4.7 References

Maciejowski J.M. — Multivariable Feedback Control, Addison Wesley, 1989

Francis B.A. — A Course in Linear H-infinity Control Theory, Springer-Verlag, 1987

Doyle J.C., Francis B.A., Tannenbaum A.R. — Feedback Control Theory, Macmillan, 1992

Braae M. — A robot arm for a first course in control engineering, IEEE Transactions on education, Vol 39 No. 1, February 1996

5 Robust Parametric Control

5.1 Introduction

This section deals with robustness analysis of the controller generated for the Robot Arm plant in section 4. It will begin with a description of the theory involved in this analysis and then proceed to present the calculations for the specific case involving the controlled linearised robot arm plant in continuous time.

The robustness of interest is with regards to controller coefficient perturbations. The aim is to find a measure that will indicate the maximum size perturbation of controller coefficients that will still stabilize the closed loop controlled system. If the controller can withstand large perturbations then the designer has more freedom for tweaking the parameters in practice. However, if very small perturbations destabilize the closed loop system, then the designer will be forced to implement the controller very accurately or ignore the controller for practical implementation.

The measure that has been used relates to the norm of a perturbation vector in parameter space. For the purposes of this section:

S = the stable region of interest

∂S = boundary of the region S

C = the complex plane

U = the area outside region S (U = C – S)

Furthermore,

$$S \cup \partial S \cup U^0 = C \quad \text{and} \quad S \cap U^0 = S \cap \partial S = \partial S \cap U^0 = \varnothing$$

5.2 The Boundary Crossing Theorem

The Boundary Crossing Theorem (Robust Control: The Parametric Approach, 1995) forms the basis for much of the theory development in this section.

Consider a family of polynomials

$$P(\lambda, s) = p_0(\lambda) + p_1(\lambda) \cdot s + p_2(\lambda) \cdot s^2 + \ldots + p_n(\lambda) \cdot s^n$$

The following observations can be made about this family:

- The family is of fixed degree n (invariant degree)
- The family is continuous with respect to λ on a fixed interval I = [a,b]

Under these assumptions the following theorem can be stated:

Theorem 1 (Boundary Crossing Theorem)

> Suppose that P(a,s) has all its roots in some region S, whereas P(b,s) has at least one root in region U. Then there exists at least one ρ in (a,b] such that:
>
> - P(ρ,s) has all its roots in $S \cup \partial S$
> - P(ρ,s) has at least one root in ∂S
>
> Where ∂S denotes the boundary of the region S
>
> A proof of this theorem can be found in Appendix 9.3.1.

Intuitively, what this theorem states is that in going from one open set to another open set disjoint from the first, the root set of a continuous family of polynomials of fixed degree must intersect at some intermediate stage the boundary of the first open set. If the family of polynomials loses degree on the interval [a,b], then the Boundary Crossing Theorem does not hold.

5.3 The Parametric Stability Margin

The parametric stability margin is a measure of the robustness of a closed loop system with respect to parametric uncertainty around some nominal point. The parameters that are investigated can be the plant coefficients, the controller coefficients or a combination of both. This measure will provide a means of comparing proposed controllers for a system as well as for identifying fragile controllers. It is this latter task that will be concentrated on in this section.

The theory presented here is completely general but will be presented for the specific case of controller parameter uncertainty. This implies that only the robustness of the controller in the face of changes to controller coefficients will be measured.

The stability region of interest in this section will be denoted by S where $S \subset C$.

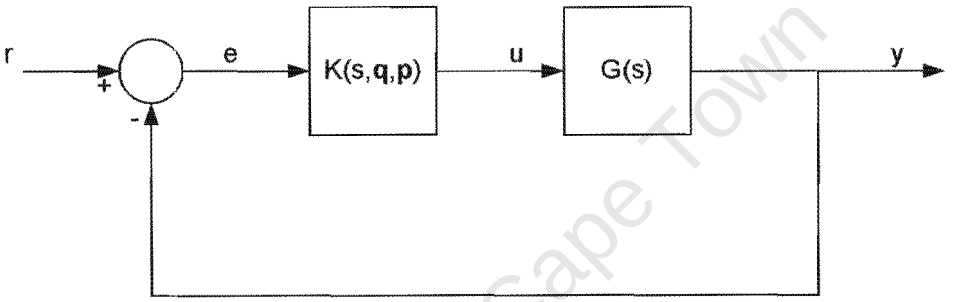


Figure 5.1 A unity feedback SISO control loop for controller parameter uncertainty

Consider first the general case. Let **P** be a vector of real parameters:

$$\mathbf{p} = [p_1, p_2, \ldots, p_l]^T$$

The characteristic polynomial for the closed loop system can then be denoted by:

$$\delta(s, \mathbf{p}) = \delta_n(\mathbf{p}) \cdot s^n + \delta_{n-1}(\mathbf{p}) \cdot s^{n-1} + \ldots + \delta_1(\mathbf{p}) \cdot s^1 + \delta_0(\mathbf{p}) \cdot s^0$$

This polynomial is therefore a real polynomial with coefficients that depend continuously on the real parameter vector **p**. For the nominal parameter $\mathbf{p}=\mathbf{p}^0$, $\delta(s, \mathbf{p}^0) = \delta^0(s)$ is stable with respect to region S (all the roots lie in S).

The perturbation in the parameter **p** from its nominal value $\mathbf{p}^0$ can be denoted by:

$$\Delta \mathbf{p} = \mathbf{p} - \mathbf{p}^0 = \left[p_1 - p_1^0, p_2 - p_2^0, p_3 - p_3^0, \ldots, p_l - p_l^0\right]$$

Now if a norm is introduced in the space of the parameters **p**, an open ball of radius ρ can be introduced:

$$\beta\left(\rho, \mathbf{p}^0\right) = \left\{\mathbf{p} : \left\|\mathbf{p} - \mathbf{p}^0\right\| < \rho\right\}$$

The hypersphere of radius ρ can be defined by

$$S\left(\rho, \mathbf{p}^0\right) = \left\{\mathbf{p} : \left\|\mathbf{p} - \mathbf{p}^0\right\| = \rho\right\}$$

With the ball β we associate the family of uncertain polynomials

$$\Delta_\rho(s) := \left\{\delta\left(s, \mathbf{p}^0 + \Delta\mathbf{p}\right) : \|\Delta\mathbf{p}\| < \rho\right\}$$

The real parametric stability margin in parameter space can then be defined as the radius, denoted $\rho^*(\mathbf{p}^0)$ of the largest ball centered at $\mathbf{p}^0$ for which $\delta(s, \mathbf{p})$ remains stable whenever $\mathbf{p} \in \beta\left(\rho^*\left(\mathbf{p}^0\right), \mathbf{p}^0\right)$.

This implies that the characteristic polynomial is stable for all perturbations with a norm less than the real parametric stability margin.

A more formal statement of this margin is given in Theorem 2.

Theorem 2 (The parametric stability margin)

a) There exists a largest stability ball $\beta\left(\rho^*, \mathbf{p}^0\right)$ centered at $\mathbf{p}^0$ with the property that:

   1. For every **p'** within the ball, the characteristic polynomial $\delta(s, \mathbf{p'})$ is stable and of degree n

   2. At least one point **p''** on the hypersphere $S(\rho^*, \mathbf{p}^0)$ itself is such that $\delta(s, \mathbf{p''})$ is unstable or of degree less than n.

b) Moreover if **p''** is any point on the hypersphere $S(\rho^*, \mathbf{p}^0)$ such that $\delta(s, \mathbf{p''})$ is unstable, then the unstable roots of $\delta(s, \mathbf{p''})$ can only be on the stability boundary.

A proof of this theorem is contained in Appendix 9.3.2.

At this point it is important to define the L_2 norm of a vector, as this is the norm that will be used to measure the size of a perturbation.

The unweighted L_2 norm is defined as follows:

$$\|\Delta \mathbf{p}\|_2 = \sqrt{\sum_{i=1}^{l} \Delta p_i^2}$$

The next section will present the above theory for the specific case where the parameters are the controller coefficients.

In Figure 5.1, the controller K has nominal coefficients that have been grouped together in vectors **q** and **p**. The controller can therefore be expressed as

$$K(s,\mathbf{q}^0,\mathbf{p}^0) = \frac{q_n^0 \cdot s^n + q_{n-1}^0 \cdot s^{n-1} + \ldots + q_0^0 \cdot s^0}{p_n^0 \cdot s^n + p_{n-1}^0 \cdot s^{n-1} + \ldots + p_0^0 \cdot s^0}$$

The 0 superscript denotes the nominal controller coefficients.

If the controller and the plant are separated into numerator and denominator, then the closed loop characteristic polynomial can be expressed as the sum of the products of the numerators and denominators:

$$\text{If } \mathrm{G(s)} = \frac{\mathrm{Gn}}{\mathrm{Gd}} \text{ and } \mathrm{K(s)} = \frac{\mathrm{Kn}}{\mathrm{Kd}}$$
then the characteristic polynomial denoted by δ
can be expressed as :
$$\delta(s) = Gn \cdot Kn + Gd \cdot Kd$$

Construct a vector of real parameters

$$dK = [dK_1, dK_2, \ldots, dK_{l-1}, dK_l]^T$$

The characteristic polynomial can then be expressed as follows:

$$\delta(s,\mathbf{dK}) = \delta_n(\mathbf{dK}) \cdot s^n + \delta_{n-1}(\mathbf{dK}) \cdot s^{n-1} + \ldots + \delta_0(\mathbf{dK})$$

The characteristic polynomial is therefore a real polynomial with coefficients that depend continuously on the real parameter vector **dK**. We suppose that for the nominal parameter $\mathbf{dK}=\mathbf{dK}^0$, $\delta(s,\mathbf{dK}^0)= \delta^0(s)$ is stable with respect to region S. This implies that the nominal system has its roots in region S.

The difference between this presentation and that presented before, is that each term in the characteristic polynomial is no longer a single parameter, but rather a value that is affected by the controller coefficient. Thus by changing the controller coefficients, the parameters in the characteristic polynomial will change, but this change will not be an identical one in the two cases.

The perturbation in the parameter **dK** from its nominal value $\mathbf{dK}^0$ can then be expressed as

$$\Delta\mathbf{dK} = \mathbf{dK} - \mathbf{dK}^0 := \left[dK_1 - dK_1^0, dK_2 - dK_2^0, \ldots, dK_l - dK_l^0\right]$$

Once again a stability ball of radius ρ will be introduced

$$\beta\left(\rho, \mathbf{dK}^0\right) = \left\{\mathbf{dK} : \left\|\mathbf{dK} - \mathbf{dK}^0\right\| < \rho\right\}$$

The hypersphere of radius ρ can be defined by

$$S\left(\rho, \mathbf{dK}^0\right) = \left\{\mathbf{dK} : \left\|\mathbf{dK} - \mathbf{dK}^0\right\| = \rho\right\}$$

With the ball β we associate the family of uncertain polynomials

$$\Delta_\rho(s) := \left\{\delta\left(s, \mathbf{dK}^0 + \mathbf{dK}\right) : \left\|\mathbf{dK}\right\| < \rho\right\}$$

The radius ρ will then give the maximum radius in parameter space of the perturbation of controller coefficients that will stabilize the closed loop system. The definition of the parametric stability margin in this case remains the same as in Theorem 2. As mentioned before the only difference is that the parameters in the characteristic polynomial are no longer affected directly, but rather indirectly by the parameters of the controller.

It is important to note that the parametric stability margin as presented here accounts for interdependent perturbations among the polynomial coefficients. The coefficients of the characteristic polynomial are not allowed to perturb independently of each. This means that the perturbation in the characteristic polynomial must be the result of a linear perturbation in the plant or controller model that leads to a change in the characteristic polynomial coefficients. This model is quite accurate for control systems where the coefficients of the characteristic polynomial are in fact dependent on other models within the system, such as the plant model and the controller model.

Now that the stability margin has been defined it is necessary to be able to calculate this margin for a system.

5.4 Stability margin computation

5.4.1 Introduction

The general case of calculating this margin involves evaluating the roots of the characteristic polynomial at each point on the stability boundary of interest. This problem is in general highly non-linear. Only the computation in the linear case will be presented here, as this is the calculation that is most applicable to the theory presented in the previous section. A detailed explanation of the image set approach for calculating this margin in the general case can be found in Robust Control: The Parametric Approach (1995).

5.4.2 The linear case

An overview of the calculation of the l_2 parametric stability margin will be presented here for the linear case. Linear implies that the coefficients of the characteristic polynomial depend linearly on the uncertain parameters. A more explicit explanation of where the calculation comes from can be found in Robust Control: The Parametric Approach (1995).

Define the perturbation vector as

$$\mathbf{p} = [p_1, p_2, \ldots, p_{l-1}, p_l]$$

In the linear cases the characteristic polynomial can then be expressed as:

$$\delta(s, \mathbf{p}) = a_1(s)p_1 + \cdots + a_l(s)p_l + b(s)$$

As $\mathbf{p} = \mathbf{p}^0 + \Delta\mathbf{p}$ the characteristic polynomial can be re-expressed as:

$$\delta(s, \mathbf{p}^0 + \Delta\mathbf{p}) = \delta(s, \mathbf{p}^0) + a_1(s)\Delta p_1 + \cdots + a_l(s)\Delta p_l \quad \text{- 5.4.2-1}$$

Assuming that s^* is a root of the above equation and that this root lies on the stability boundary, then using Matrix-Vector notation, the expression can be restated as follows:

$$\begin{aligned} \mathbf{A}(s_r)\cdot\mathbf{t}(s_r) &= b(s_r) \quad \text{for the real case where} \quad s^* = s_r \\ \mathbf{A}(s_c)\cdot\mathbf{t}(s_c) &= \mathbf{b}(s_c) \quad \text{for the complex case where} \quad s^* = s_c \end{aligned} \quad \text{- 5.4.2-2}$$

$\mathbf{t}(s_r)$ is one dimensional vector that is equivalent to the perturbation vector. $\mathbf{A}(s_r)$ is a one-dimensional vector of the coefficients of the perturbation vector elements as derived from expression 5.4.2-1. $b(s_r)$ is a scalar value.

The complex case is almost identical to the real case, except that there is an expression for the real part and the imaginary part of the root. Hence $\mathbf{t}(s_c)$ is one dimensional vector that is equivalent to the perturbation vector. $\mathbf{A}(s_c)$ is a matrix with 2 rows and L columns corresponding to the coefficients of the perturbation vector elements as derived from expression 5.4.2-1. $\mathbf{b}(s_c)$ is a 2 element vector.

The expressions above calculate the perturbation vector that will produce a root on the stability boundary. The calculation to determine the perturbation vector that will result in a drop of degree is calculated from:

$$\delta_n\left(s, \mathbf{p}^0 + \Delta\mathbf{p}\right) = 0$$

This expression simply calculates when the highest power coefficient drops to zero.

In matrix-vector form this expression can be stated as:

$$\mathbf{A}_n \cdot \mathbf{t}_n = b_n \qquad - \qquad 5.4.2\text{-}3$$

where $\mathbf{A}_n$ is a one-dimensional vector of coefficients of the perturbation vector elements. $\mathbf{t}_n$ is the perturbation vector and b_n is a scalar.

Let $t^*(s_c)$, $t^*(s_r)$ and t^*_n denote the minimum norm solutions of the matrix vector equations presented above. Then for any norm

$$\left\|t^*\left(s_c\right)\right\| = \rho\left(s_c\right)$$
$$\left\|t^*\left(s_r\right)\right\| = \rho\left(s_r\right)$$
$$\left\|t^*{}_n\right\| = \rho_d$$

The minimum of these norms then provides the parametric stability margin for the system in question. The principle is to sweep over the stability boundary and solve for the t vector for each point on the boundary. By calculating the norms of these vectors and taking the minimum, you will then have the minimum distance to instability as a norm.

The solutions to the matrix-vector equations will now be presented for the case of the L_2 norm.

5.4.3 *The l_2 Parametric Stability Margin*

Suppose that the length of the perturbation vector $\Delta\mathbf{p}$ is measured by the unweighted L_2-norm. The minimum norm solutions to the matrix-vector equations 5.4.2-2 and 5.4.2-3 are desired.

If it is assumed that $\mathbf{A}(s_c)$ has full row rank (ie. a rank of 2) then

$$t^*(s_c) = \mathbf{A}^T(s_c)\left[\mathbf{A}(s_c)\mathbf{A}^T(s_c)\right]^{-1}\mathbf{b}(s_c)$$

Similarly in the real case

$$t^*(s_r) = \mathbf{A}^T(s_r)\left[\mathbf{A}(s_r)\mathbf{A}^T(s_r)\right]^{-1}b(s_r)$$

and

$$t_n^* = \mathbf{A}_n^T\left[\mathbf{A}_n\mathbf{A}_n^T\right]^{-1}b_n$$

The real case solutions assume that $\mathbf{A}(s_r)$ and $\mathbf{A}_n$ are non-zero vectors.

If $\mathbf{A}(s_c)$ does not have full row rank then two cases can occur.

1. If the rank is equal to zero, then the equation is inconsistent. Hence the matrix-vector equation does not have a solution and $\rho(s_c) = \infty$.

2. If the rank is equal to one, then the matrix-vector equation will only be consistent if

$$rank\left[\mathbf{A}(s_c), \mathbf{b}(s_c)\right] = 1$$

   If this rank condition holds then the two equations for the complex case of 5.4.2-2 can be replaced by a single equation (as in the real case) and the minimum norm solution to this equation is then solved for in the same fashion as shown above.

   However, if this rank condition does not hold, then the equation cannot be satisfied and $\rho(s_c) = \infty$.

In finding the parametric stability margin, it is necessary to sweep the solution s* over a particular range on the stability boundary. At each point along this sweep, the matrix-vector equations are solved to find the L_2 norm. In this way the minimum distance to the stability boundary is found within a specific range of the stability boundary.

These calculations will be made clearer when they are performed on the specific case of the robot arm plant.

5.5 References

Bhattacharyya S.P.,
Chapellat H., Keel L.H. Robust Control: The Parametric Approach, Prentice Hall, 1995

6 Robot Arm Plant: Stability margin analysis

6.1.1 Introduction

This section will analyze the robot arm plant using the L_2 stability margin calculation. These calculations were performed using a routine written in Matlab as the sizes of the matrix-vector equations were rather large for manual computation.

In order to verify the routines, the quoted examples of Keel et al were used as test samples. Example 1 and 3 were analyzed using the Matlab routines and the results were then compared to the quoted results.

6.1.2 Test Case 1: Example 1

A matlab routine called eg1 was created to evaluate the parametric stability margin for Example 1 in the paper of Keel et al. The general format for use of this routine is:

```
function [freqr,rhowr,perturbsr]=eg1(Sfreq,Efreq,Stepfreq)
```

where

freqr = returned frequency values over which the routine sweeps
rhowr = the l2 norms at each frequency step
peturbsr = the perturbation vector at each frequency step

Sfreq = start frequency for the sweep
Efreq = end frequency for the sweep
Stepfreq = size of the step for sweep from Sfreq to Efreq

Referring to section 5.4.3, this routine sweeps s_c from Sfreq to Efreq and calculates the norms at each point along the sweep. In effect the calculation is finding the shortest distance to the imaginary axis (stability boundary) at each point along the sweeping range. The shortest distance would then be the parametric stability margin.

The frequency range in the routine allows the user to obtain more accuracy around certain areas of interest. For example, an initial sweep with a step of 5 rad/s could be performed to find a minimum area. This area could then be sweeped with a finer step size to isolate the minimum norm solution.

The plant in example 1 is given by:

$$P(s)=\frac{s-1}{s^2-s-2}$$

The nominal controller is given by:

$$C(s)=\frac{q_6^0 s^6+q_5^0 s^5+q_4^0 s^4+q_3^0 s^3+q_2^0 s^2+q_1^0 s^1+q_0^0 s^0}{p_6^0 s^6+p_5^0 s^5+p_4^0 s^4+p_3^0 s^3+p_2^0 s^2+p_1^0 s^1+p_0^0 s^0}$$

where

Q_6^0	379.000000000000000000	P_6^0	3.000000000000000000
Q_5^0	39383.000000000000000000	P_5^0	-328.000000000000000000
Q_4^0	192306.000000000000000000	P_4^0	-38048.000000000000000000
Q_3^0	382993.000000000000000000	P_3^0	-179760.000000000000000000
Q_2^0	383284.000000000000000000	P_2^0	-314330.000000000000000000
Q_1^0	192175.000000000000000000	P_1^0	-239911.000000000000000000
Q_0^0	38582.000000000000000000	P_0^0	-67626.000000000000000000

In Example 1, it is known that the solution lies at a frequency of 14.27 rad/s. Hence the routine can be run around this range.

Initially eg1 was called with Sfreq=14, Efreq=14.5 and Stepfreq=0.01. This returned rho=0.15825936999449. This is not identical to the published value so the routine was rerun with a finer step size.

By sweeping with a smaller step size, rho = 0.158139031 at a frequency of 14.2716741239 rad/s. This value agrees with the published value up to 9 decimal places. This indicates that the routine is functioning properly.

It is important to consider the destabilizing perturbation in this case.

Parameters	Nominal	Perturbation
q6	379.00000000000000000000	0.00028581100002611500000000000
q5	39383.00000000000000000000	-0.00076885899761691700000000000
q4	192306.00000000000000000000	-0.00000140300835482776000000000
q3	382993.00000000000000000000	0.00000377499964088202000000000
q2	383284.00000000000000000000	0.00000000698491930961609000000
q1	192175.00000000000000000000	-0.00000001798616722226140000000
q0	38582.00000000000000000000	0.00000000000000000000000000000
p6	3.00000000000000000000	0.15813474800000000000000000000
p5	-328.00000000000000000000	0.00028109099997664100000000000
p4	-38048.00000000000000000000	-0.00077638599759666200000000000
p3	-179760.00000000000000000000	-0.00000137998722493649000000000
p2	-314330.00000000000000000000	0.00000381201971322298000000000
p1	-239911.00000000000000000000	0.00000000701402314007282000000
p0	-67626.00000000000000000000	-0.00000001800071913748980000000
L2 Norms	751823.2201	0.158139031

All the perturbations are relatively small in relation to the original coefficient. If one perturbation is increased slightly, then the closed loop system goes unstable. However, if the perturbation on the coefficient with the largest relative perturbation (p^6) is reduced, then the other coefficients can be perturbed to a much larger degree. The system remains stable even though the L_2 norm of the perturbation is greater than the parametric stability margin. This therefore contradicts the parametric stability margin. These results are shown below, where the p^6 perturbation was reduced and p^5 and p^2 was increased.

Parameters	Nominal	Perturbation	Percentage Perturbation
q6	379.00000000000000000000	0.00028581100002611500000000000	0.000075411873358
q5	39383.00000000000000000000	-0.00076885899761691700000000000	-0.000001952261122
q4	192306.00000000000000000000	-0.00000140300835482776000000000	-0.000000000729571
q3	382993.00000000000000000000	0.00000377499964088202000000000	0.000000000985658
q2	383284.00000000000000000000	0.00000000698491930961609000000	0.000000000001822
q1	192175.00000000000000000000	-0.00000001798616722226140000000	-0.000000000009359
q0	38582.00000000000000000000	0.00000000000000000000000000000	0.000000000000000
p6	3.00000000000000000000	0.11113474800000000000000000000	3.704491600000000
p5	-328.00000000000000000000	1.30028109099990000000000000000	-0.396427161890213
p4	-38048.00000000000000000000	-0.00077638599759666200000000000	0.000002040543518
p3	-179760.00000000000000000000	-0.00000137998722493649000000000	0.000000000767683
p2	-314330.00000000000000000000	1000.00020081200000000000000000000	-0.318137053673528
p1	-239911.00000000000000000000	0.00000000701402314007282000000	-0.000000000002924
p0	-67626.00000000000000000000	-0.00000001800071913748980000000	0.000000000026618
L2 Norms	751823.2201	1000.001052	0.001330101

The closed loop poles in this situation are indicated in the following table.

Closed loop poles	
Real Part	Imaginary Part
-0.0389145559	-14.0191928871
-0.0389145559	14.0191928871
-5.8954924471	-11.1977103688
-5.8954924471	11.1977103688
-1.0984428566	-0.3441106774
-1.0984428566	0.3441106774
-1.0000007005	0.0000000000
-0.7450683636	0.0000000000
-1.4830792631	0.8356215888

Despite the fact that this perturbation exceeds the parametric stability margin in norm size, the relative size of the perturbations per coefficient must still remain rather small to ensure closed loop stability. Therefore, this controller is still fragile as it cannot withstand a 10% perturbation on the higher power coefficients.

The next section will consider Example 3 in the paper.

6.1.3 Test Case 2: Example 3

The Matlab routine for example 3 has an identical format to Example 1 except that the routine is called eg2.

In this case the routine can be called with any frequency range, as the minimum is actually at the origin. If eg2 is called with Sfreq=1, Efreq=10 and Stepfreq=1, then rho = 8.944271909999159 at the origin. This value agrees with the published value for all decimal places.

The two cases above therefore indicate the matlab routines for calculating the parametric stability margin are performing correctly.

It is now possible to start analyzing the robot arm using similar routines as in sections 6.1.2 – 6.1.3

6.1.4 *The Robot Arm Plant*

The parametric stability analysis was performed on the open loop unstable plant using the continuous time controller as presented in section 4.5.

The matlab routine to calculate the parametric stability margin for the original open loop unstable plant is RobotAL2.

Running RobotAL2 returns a parametric stability margin of

1.549768958908020e+004

at a frequency of

1.421000000000000e+001 rad/s

However, running RobotAL2 up to frequencies of 500 rad/s, the minimum norm was located at 173.85 rad/s. The minimum norm solution is rho = 18.73769664.

The normalized ratio of change in controller coefficients to destabilize the closed loop system is

$$\frac{\rho}{\left\|\mathbf{p}^0\right\|_2} = \frac{18.73769664}{24832734896} = 7.54556303055615\text{E}-10$$

This implies that a normalized change in controller coefficients of less that 1 part in a billion will destabilize the closed loop system. This would indicate that the controller designed is not robust, but rather very fragile.

It is now necessary to investigate this result more closely by constructing a destabilizing perturbation to add to the nominal controller.

The perturbation that yielded the minimum norm solution is given in the table below.

Parameters	Nominal	Perturbation
q6	-0.000044867468030229 60	-0.018960101174245400000000000000
q5	453.766130704534000000 00	0.000159882702276172000000000000
q4	30543021.935533700000000000000	0.000000627323071866419000000000
q3	642725722.246738000000000000000	-0.000000005289956367238860000000
q2	4491215385.045620000000000000000	-0.000000000020755914374047800000
q1	13168465210.152500000000000000000	0.000000000000175026053280946000
q0	20552318849.399900000000000000000	0.000000000000000686740215406168
p6	1.000000000000000000	18.736970921818100000000000000000
p5	2022.585993707330000000 00	-0.163817192363491000000000000000
p4	641140.081606532000000000 00	-0.000619940476484009000000000000
p3	61667185.411377300000000000000	0.000005420134808014160000000000
p2	440026902.673043000000000000000	0.000000020511650254828200000000
p1	263657243.139968000000000000000	-0.000000000179333200094533000000
p0	80308161.853313500000000000000	-0.000000000000678658374691959000
L2 Norms	24832734896	18.73769664

The poles of the closed loop system at this point are listed in the table below:

Closed Loop Poles	
Real Part	Imaginary Part
0.0000000000	-173.8500000000
0.0000000000	173.8500000000
-77.4851952371	0.0000000000
-10.5851445532	0.0000000000
-3.2801209116	-5.3842735227
-3.2801209116	5.3842735227
-5.3755916791	-1.5516113141
-5.3755916791	1.5516113141

It is thus clear that the perturbation places poles on the stability boundary (imaginary axis in the s-plane). If one of the coefficients of the perturbation is increased slightly, then the closed loop system will become unstable.

Thus everything agrees with the predictions made from the parametric stability margin calculation. However, there is one problem. Consider the perturbation on q6 and p6. The relative sizes of these perturbations are 42258.0145629590% and 1873.6970921818% respectively. This would indicate that the perturbed controller is in fact very different from the nominal controller as a result of the sizes of the perturbations. Thus although the L_2-norm of the perturbation is small, the relative size of the coefficient perturbations are very large.

In practice, the controller would be implemented as close as possible to the electronically designed controller. It should therefore be able to tolerate a certain percentage of change per controller coefficient, probably in the region of 20% - 30%. If the controller in this case was implemented with such a large perturbation on a coefficient, then it would indicate a very bad implementation. The conclusion that can be drawn from this is that the parametric stability margin is not necessarily an absolute measure of the robustness or fragility of a controller.

To test this assumption, a perturbation was applied to the controller coefficients. This perturbation is the one discussed in section 4.6.3. This shall now be analysed from a parametric stability margin aspect.

Parameters	Nominal	Perturbation
q6	-0.000044867468030229 60	-0.000089734936060459200000000000
q5	453.766130704534000000	907.532261409068000000000000000000
q4	30543021.935533700000000000000	0.000000627323071866419000000000
q3	642725722.246738000000000000000	-0.000000005289956367238860000000
q2	4491215385.045620000000000000000	-0.000000000020755914374047800000
q1	13168465210.152500000000000000000	0.000000000000175026053280946000
q0	20552318849.399900000000000000000	0.000000000000000686740215406168
p6	1.000000000000000000000	2.000000000000000000000000000000
p5	2022.585993707330000000000	-0.163817192363491000000000000000
p4	641140.081606532000000000000	-0.000619940476484009000000000000
p3	61667185.411377300000000000000	0.000005420134808014160000000000
p2	440026902.673043000000000000000	0.000000020511650254828200000000
p1	263657243.139968000000000000000	-0.000000000179333200094533000000
p0	80308161.853313500000000000000	-0.000000000000067865837469195900 0
L2 Norms	24832734896	907.53448

From the above table the main perturbations of interest are the 200% perturbation on q6 and q5 and the 300% perturbation on p6. The rest of the perturbations are very small percentages of the original coefficient. The first point to note is that the norm of the perturbation is 907.53448, which is already greater than the parametric stability margin of 18.737696. The poles of the closed loop system however, turn out to be stable and the performance of the closed loop system is very similar to the unperturbed case. This further strengthens the conclusion that the parametric stability margin is not necessarily an absolute measure of robustness.

The poles for the perturbed system given above are shown in the table below for completeness.

Closed loop poles	
Real Part	Imaginary Part
-268.0978239182	-254.4154787621
-268.0978239182	254.4154787621
-112.9872832173	0.0000000000
-10.5461487535	0.0000000000
-3.2816731045	-5.3841757566
-3.2816731045	5.3841757566
-5.3806714835	-1.5511731418
-5.3806714835	1.5511731418

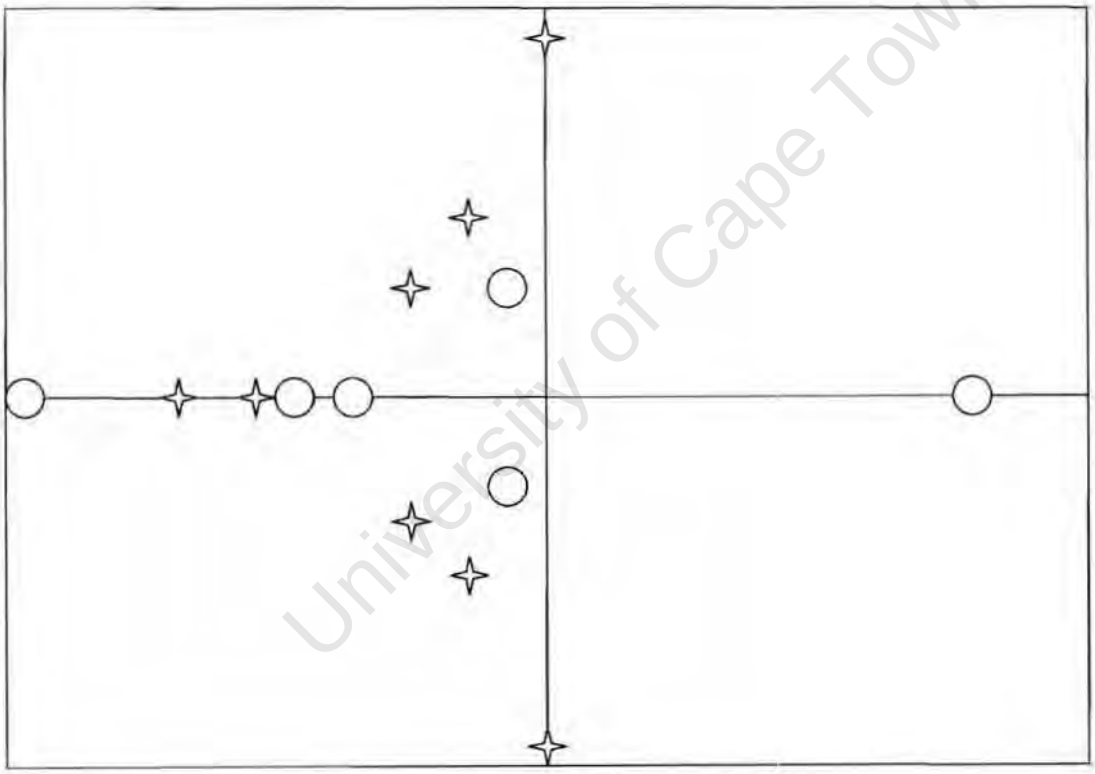
It is also possible to obtain a destabilizing perturbation with a smaller norm than the calculated one. However, this minimum turns out to be invalid as it violates the Boundary Crossing Theorem. This case shall be investigated next.

Parameters	Nominal	Perturbation
q6	-0.00004486746803022960	-0.018960101174245400000000000000
q5	453.76613070453400000000	0.000159882702276172000000000000
q4	30543021.935533700000000000000	0.000000627323071866419000000000
q3	642725722.24673800000000000000	-0.000000005289956367238860000000
q2	4491215385.04562000000000000000	-0.000000000020755914374047800000
q1	13168465210.15250000000000000000	0.000000000001750260532809460000
q0	20552318849.39990000000000000000	0.000000000000006867402154061680
p6	1.00000000000000000000	-1.000000100000000000000000000000
p5	2022.58599370733000000000	-0.163817192363491000000000000000
p4	641140.08160653200000000000	-0.000619940476484009000000000000
p3	61667185.41137730000000000000	0.000005420134808014160000000000
p2	440026902.67304300000000000000	0.000000020511650254828200000000
p1	263657243.13996800000000000000	-0.000000000179333200094533000000
p0	80308161.85331350000000000000	-0.000000000006786583746919590000
L2 Norms	24832734896	1.013506866

In the table, the perturbation on p6 has been changed to –1.0000001. The parametric stability margin has now dropped to 1.013506866. However, the system is already unstable at this point as can be seen from the closed loop poles in the table below.

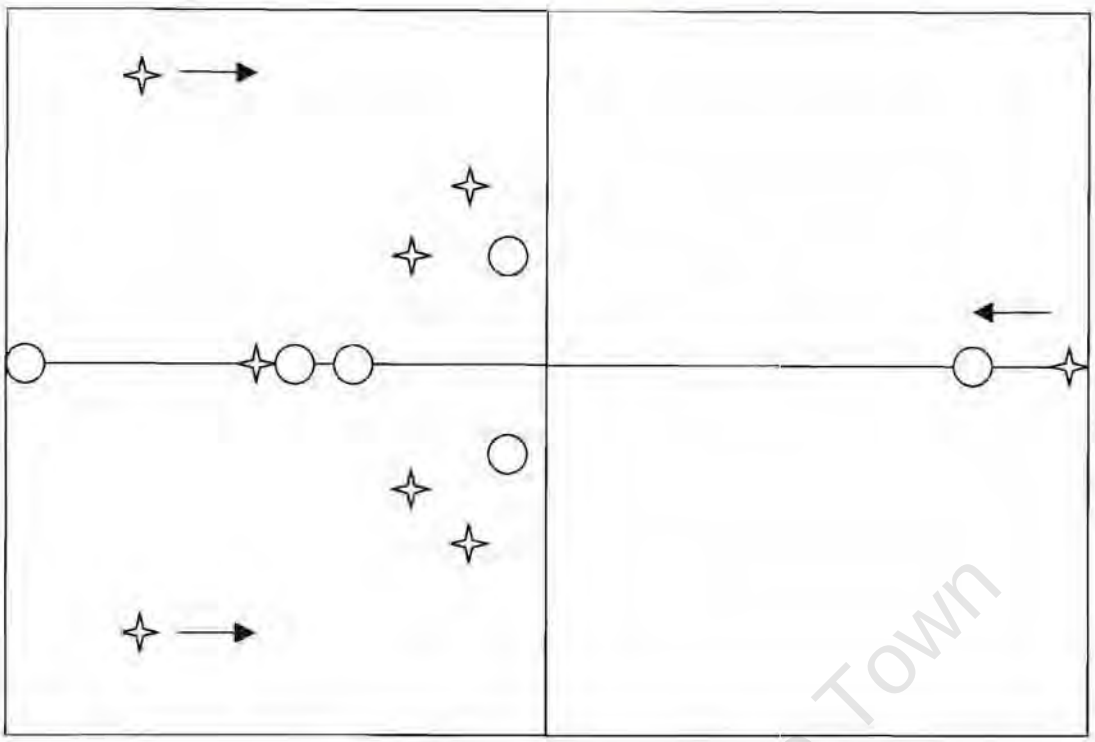
Closed Loop Poles	
Real Part	Imaginary Part
20224222070.3571000000	0.0000000000
-146.0193368306	-39.4737374117
-146.0193368306	39.4737374117
-10.5826663853	0.0000000000
-3.2797426054	-5.3834321771
-3.2797426054	5.3834321771
-5.3739402332	-1.5515717278
-5.3739402332	1.5515717278

The relative locations of the poles and zeroes in the original case where rho=18.73 is shown below.



The stars indicate the poles while the circles indicate the zeroes.

Now consider the case where p6=-1.0000001



Adjusting p6 does not change the zero positions. The arrows in the above drawing shows the direction in which the poles move as p6 is decreased from –1.1 to –10 to -100, etc. The other poles only move slightly, but it is the 3 poles with arrows that do the major movement. Basically as p6 changes the 3 poles approach the imaginary axis. It is in fact the two imaginary poles that reach the imaginary axis first. Their crossing corresponds to the other minimum quoted above, ie: 15497 at a frequency of 14.21 rad/s.

The reason why this type of perturbation is invalid is illustrated by the appearance of the rightmost pole. Basically, it has passed from the left half plane to the right half plane without crossing the boundary. (ie. it violates the Boundary Crossing Theorem). This type of change also violates the principle of the continuity of roots as their coefficients change. This change has thus created a totally new system that is very different from the original plant, which only had stable poles that moved towards the right half plane across the imaginary axis.

If continuity is considered, then to reach –1.0000001, implies that the p6 coefficient would have to reach zero at some point. This would result in a loss of degree in the final characteristic polynomial. This loss of degree is a violation of the Boundary Crossing Theorem. For this reason, this particular case should be ignored.

6.1.5 *A second test case*

In order to verfiy the results that were generated for the robot arm plant, a second test case was chosen using a very simple plant.

The plant model is given below.

$$G(s) = \frac{1}{s-1}$$

Using an H-infinity design method, a controller was generated to stabilize the system. The design criteria was for the plant to track step inputs with minimal error. The quality of the output response was not critical. The main issue was to generate a stable response.

The coefficients of the controller are shown in the table below.

	Controller Numerator	Controller Denominator
s5	0.000000159999999700	1.000000000000000000
s4	425.139016798282000000	10220.025650291000000000
s3	12759360.343579400000000000	2221036.539317940000000000
s2	2177265816.247520000000000000	130015139.095807000000000000
s1	15397169621.079200000000000000	1798865 08.373335000000000000
s0	13390582022.463400000000000000	126912699.221052000000000000

The step response for the plant using this controller is shown in Figure 6.1. The settling time is about 0.7 seconds with approximately 20% overshoot. However, the response is stable. Hence the generated controller is satisfactory. The next step is to perform the parametric stability analysis as was done on the robot arm plant.

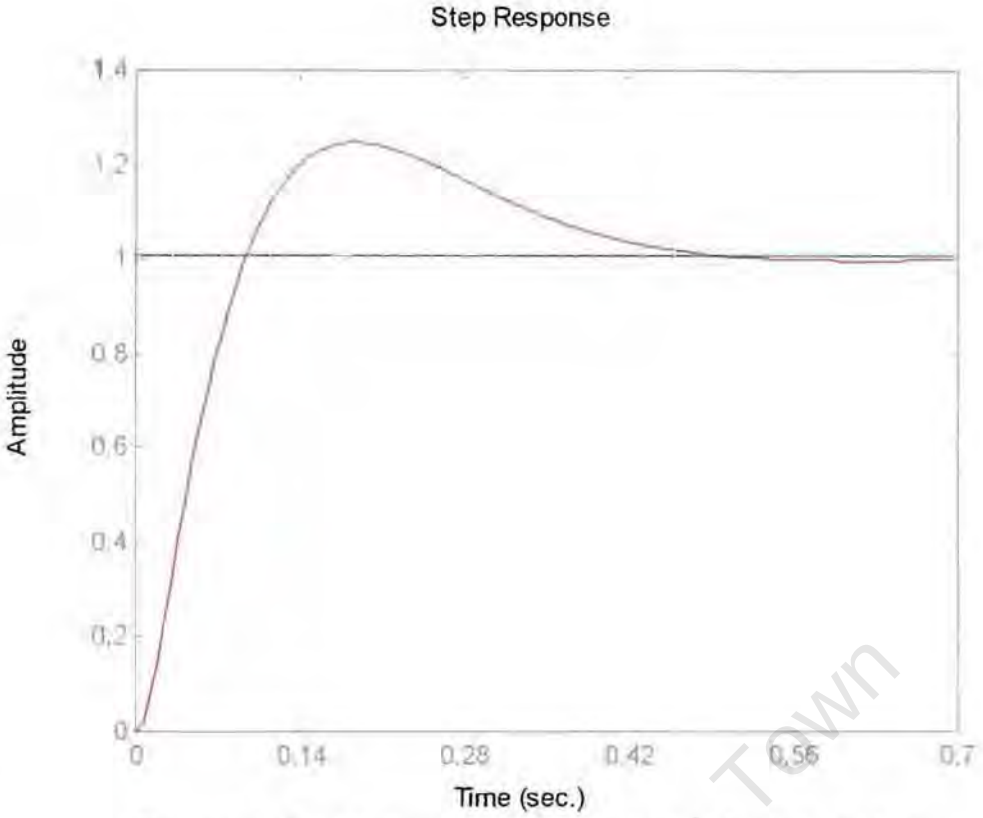


Figure 6.1: Closed loop step response of the simple test plant

The matlab routine to calculate the parametric stability margin is TestPAL25. After running TestPAL25 on the closed loop system the parametric stability margin was

147.2265312

at a frequency of

117.7 rad/s

The normalized change in controller coefficients to destabilize the closed loop system is therefore

$$\frac{\rho}{\left\|\mathbf{p}^0\right\|_2} = \frac{147.2265312}{20522829808} = 7.173792921218382e-009$$

This implies that a change in controller coefficients of less than 1 part in a billion is enough to destabilize the closed loop system.

The table below shows the nominal controller coefficients together with the perturbation that generates the minimum norm solution.

	Nominal Controller Coefficients	Perturbation
q5	0.000000159999999700	-3.058246400488674000000000000000000
q4	425.13901679828200000000	-0.010402219187383800000000000000000
q3	12759360.34357940000000000000	0.000220759574432264000000000000000
q2	2177265816.24752000000000000000	0.000000750884388284936000000000
q1	15397169621.07920000000000000000	-0.000000015935534045144800000000
q0	13390582022.46340000000000000000	-0.000000000054202603734198600000
p5	1.000000000000000000	147.16320545127900000000000000000000
p4	10220.025650291000000000	-3.047844185699360000000000000000000
p3	2223036.539317940000000000	-0.010622978761816000000000000000000
p2	130015139.095807000000000000	0.000220008690043979000000000000
p1	179886508.373335000000000000	0.000000768199223300810000000000
p0	126912699.221052000000000000	-0.000000015881331441410600000000
L2 Norms	20522829808	147.2265312

If the relative size of the perturbation is considered with respect to the nominal coefficient, it is once again noticed that q5 and p5 have relatively large perturbations: 1911404003% and 14716% respectively. Thus once again, as with the robot arm plant, the destabilizing perturbation is actually rather large if the relative size of the perturbation is considered.

A solution that is limited to less than 100% of each nominal coefficient will not actually result in instability. Thus the parametric stability margin does not accurately reflect the fragility of the controller.

This is yet another simple example that uses a full H-infinity design and does not agree with the results of the parametric stability tests.

6.2 References

Bhattacharyya S.P., Chapellat H., Keel L.H. — Robust Control: The Parametric Approach, Prentice Hall, 1995

Braae M. — A robot arm for a first course in control engineering, IEEE Transactions on education, Vol 39 No. 1, February 1996

Bhattacharyya S.P., Keel L.H. — "Robust, Fragile, or Optimal?", IEEE Transactions on Automatic Control, Vol 42 No. 8, August 1997

7 Conclusions

This project began as an application of the parametric stability margin analysis on a real plant to verify the results presented by Keel at al in their paper (1997). What it led to were results that actually contradict their theory in a number of ways.

The main conclusions are:

- A small parametric stability margin does not necessarily imply a fragile controller. The control engineer should consider the relative size of the destabilizing perturbation per coefficient.
- Despite the definition of the parametric stability margin, it was possible to find a perturbation with a norm greater than this margin that still stabilized the physical plant. This occurred with the robot arm plant and example 1 in Keel et al's paper (1997).
- The main measure of instability is therefore to consider the maximum size perturbation on leading coefficients in the numerator and denominator of the controller. It is these coefficients that primarily determine instability. A more accurate definition for a fragile controller is one where the maximum perturbation per coefficient is less than a certain percentage of the original coefficients. If all these percentages are less than say 1%, then the controller is fragile as there is no room for tweaking controller parameters.
- The designed controller, although predicted to be fragile, controlled the physical plant successfully, even when coefficients were perturbed beyond the parametric stability margin.

The result is therefore that case must be taken when applying the parametric stability margin analysis. The results should not be taken at face value and the control engineer should rather investigate them more closely, before discarding potentially successful controllers.

8 Bibliography

Bhattacharyya S.P., Chapellat H., Keel L.H. — Robust Control: The Parametric Approach, Prentice Hall, 1995

Bhattacharyya S.P., Keel L.H. — "Robust, Fragile, or Optimal?", IEEE Transactions on Automatic Control, Vol 42 No. 8, August 1997

Braae M. — A robot arm for a first course in control engineering, IEEE Transactions on education, Vol 39 No. 1, February 1996

Braae M. — Control Theory for Electrical Engineers, UCT Press, 1994

Braae M. — Digital and State Space Control Theory, UCT Press, 1996

Doyle J.C., Francis B.A., Tannenbaum A.R. — Feedback Control Theory, Macmillan, 1992

Francis B.A. — A Course in Linear H-infinity Control Theory, Springer-Verlag, 1987

Maciejowski J.M. — Multivariable Feedback Control, Addison Wesley, 1989

Makila, Pertti M. — "Comments on 'Robust, Fragile, or Optimal?'", IEEE Transactions on Automatic Control, Vol 43, No 9, September 1998

9 Appendices

9.1 Appendix: H-infinity theory

9.1.1 Derivation of the F_l cost function for H-infinity control

Begin with the figure 9.1 shown below.

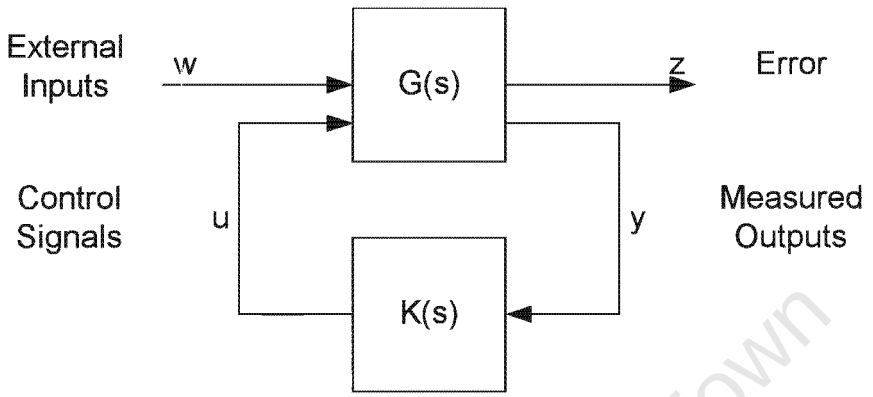


Figure 9.1 Standard representation of an uncertain plant under feedback control excluding plant uncertainty

Suppose that **G(s)** is partitioned as

$$\mathbf{G(s)} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix}$$

then

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \cdot \begin{bmatrix} w \\ u \end{bmatrix}$$

so that

$z = G_{11}w + G_{12}u$ and $y = G_{21}w + G_{22}u$

Substituting $u = Ky$

$y = G_{21}w + G_{22}Ky \Rightarrow y = (1 - G_{22}K)^{-1}G_{21}w$

Hence

$z = G_{11}w + G_{12}K(1 - G_{22}K)^{-1}G_{21}w$

which with some grouping becomes

$z = \left[G_{11} + G_{12}K(1 - G_{22}K)^{-1}G_{21}\right]w$

This expression is often stated as

$z = F_l(G, K)w$

9.1.2 *The H-infinity norm*

In order to describe the H-infinity norm, it is necessary to introduce the principal gains of a matrix. The principal gains are also called the singular values of a matrix. If we consider a matrix **G**, then the singular values are the positive square roots of the eigenvalues of the square matrix

$$\mathbf{G}\cdot\mathbf{G}^{\mathrm{H}} \quad \text{where} \quad \mathbf{G}^{\mathrm{H}} = \overline{\mathbf{G}}^{\mathrm{T}}$$ (ie. the transpose of the conjugate of matrix **G**)

These principal gains are often denoted by $\begin{pmatrix}\sigma_1 & \sigma_2 & \cdots & \sigma_m\end{pmatrix}$ where $\overline{\sigma}$ is the largest of these values and $\underline{\sigma}$ denotes the smallest value. It is important to note that the **G** matrix is a function of frequency. The principal gains are also frequency dependent.

The H-infinity norm is a frequency-independent value that is calculated from the principal gains. The H-infinity norm is expressed as

$$\|\mathbf{G}\|_{\infty} = \sup_{\omega} \overline{\sigma}(\mathbf{G}(j\omega))$$

The H-infinity norm can thus be defined as the maximum principal gain of a matrix **G** over all frequencies. This norm also represents the maximum gain of a Single Input Single Output (SISO) or Multiple Input Multiple Output (MIMO) system.

9.1.3 *The Youla Parametrization*

The Youla parametrization of all stabilizing controllers can be expressed as

$$K = (I - QG)^{-1} Q$$

This implies that

$$Q = K(I + GK)^{-1}$$

Using these subsitutions it is possible to reformulate the sensitivity function expression.

$$S = (I + GK)^{-1}$$

Substituting for K

$$S = \left(I + G(I - QG)^{-1} Q\right)^{-1}$$

$$S = \left(\frac{I - QG + GQ}{(I - QG)}\right)^{-1}$$

$$S = (I - QG)^{-1}$$

Hence the sensitivity minimization problem

$$\text{minimize} \left\| W(I + GK)^{-1} \right\|_{\infty}$$

can be re-expressed as

$$\underset{\text{stable Q}}{\text{minimize}} \left\| W(I - QG)^{-1} \right\|_{\infty}$$

The parameter Q ranges over all proper stable transfer functions. Furthermore, Q is only proper if K is proper and K is only proper if Q is proper. Hence the minimization involving Q will ensure a proper, stable K, provided that Q is stable and proper.

This parametrization is completely general and works for multivariable and single variable plants. The representation used in this dissertation is for the single variable case where the identity matrix I is a 1 x 1 matrix and Q, G and K are scalar transfer functions.

9.1.4 *Matlab routines for implementing H-infinity control*

The robust control toolbox for Matlab provides routines to perform H-infinity control. The main H-infinity routine implements a version of the 4-block Glover-Doyle algorithm using the so-called Loop Shifting Formulae. The routine takes in one augmented matrix containing the plant and weighting functions on the sensitivity function (W_1), the complementary sensitivity function (W_3) and the input function (W_2).

The Glover-Doyle algorithm has a number of rank constraints that need to be met before the routine can be run. The reader is referred to other sources on this topic such as Multivariable Feedback Design by J.M. Maciejowski. Only one of these constraints will be highlighted here as it has bearing on the robot arm plant.

If the state space realization of the plant does not have a D-term (ie. D=0) then this violates a rank condition in the augmented matrix for the Glover-Doyle algorithm. For this reason it is necessary to alter this value to some non-zero value. The general procedure for performing this change is to calculate the minimum principal gain at the upper bound of the frequency range for which the plant should be controlled.

For example, most of the control will take place at low frequencies. The system could be susceptible to high frequency disturbances up to x rad/s. Thus calculate the principal gain G(jx). This value can then be substituted for the D-term in the state space model of the plant.

The robot arm plant was initially modified in this fashion for continuous time design. At 100 rad/s, $|G(j100)| = 0.0008358$. Thus the D-term was adjusted to 0.0008. The plant response was investigated after this change was made. The responses were not changed noticeably. However, if the D-term was made a lot smaller than 0.0008, then the responses became very noisy.

The routine in Matlab is principally for continuous time design, but by doing some preprocessing on the plant it is possible to design for a digitally controlled system.

A sample set of matlab commands now follows for continuous and digital H-infinity design.

A sample set of matlab commands for a continuous time H-infinity design

Convert plant to state-space

[ag,bg,cd,dg] = tf2ss(Plant_numerator, Plant_denominator)

Adjust D-term if necessary

dg=0.0008

Specify weighting functions in transfer function notation

W1=[w1_numerator, w1_denominator]

W2=[w2_numerator, w2_denominator]

W3=[w3_numerator, w3_denominator]

Pack plant model

ss_g=mksys(ag,bg,cg,dg)

Create augmented matrix

TSS_=augtf(ss_g,W1,W2,W3)

Perform H-infinity control

[ss_cp,ss_cl,hinfo) = Hinf(TSS_, 1)

If a controller can be found, it will be contained in ss_cp. If not, an error is generated by the Hinf routine.

Digital control follows the same procedure, except that the plant is preprocessed slightly before running the routine. The generated controller will then be able to handle a certain sample time for digital control.

A sample set of matlab commands for a discrete time H-infinity design

Convert plant to state-space

[a,b,c,d] = tf2ss(Plant_numerator, Plant_denominator)

Convert plant to discrete time using a 50ms sample time

[az,bz] = c2d(a, b, 0.05)

Convert the plant back to continuous time using a bilinear transform

[ag,bg,cg,dg] = bilin(az,bz,c,d,-1,'Tustin',0.05)

Adjust D-term if necessary

dg=0.0008

Specify weighting functions in transfer function notation

W1=[w1_numerator, w1_denominator]

W2=[w2_numerator, w2_denominator]

W3=[w3_numerator, w3_denominator]

Pack plant model

ss_g=mksys(ag,bg,cg,dg)

Create augmented matrix

TSS_=augtf(ss_g,W1,W2,W3)

Perform H-infinity control

[ss_cp,ss_cl,hinfo) = Hinf(TSS_, 1)

9.2 Appendix: Robot Arm physics

This appendix deals with some aspects of the model derivation for the robot arm plant as well as initial H-infinity controller attempts using different methods.

9.2.1 Robot Arm Model

The robot arm consists of a single 9V electric motor to which a shaft is attached. A weight is attached to the other end of this shaft.

A block diagram of the robot arm is shown in Figure 9.2.

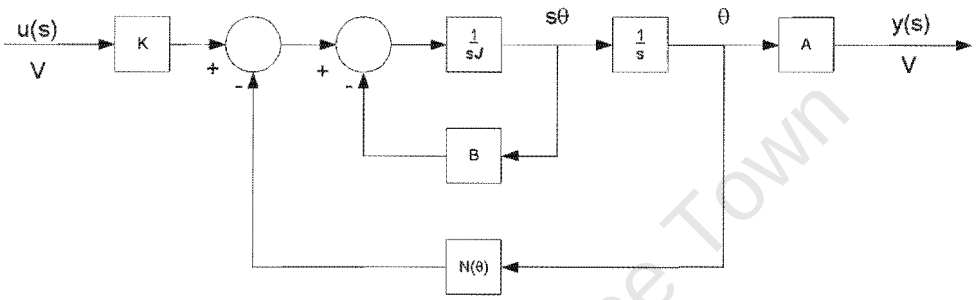


Figure 9.2 Block diagram of the robot arm

A brief description of this block diagram follows:

- A voltage is inputted into the controller (u(s))
- The controller outputs a voltage to the motor
- This voltage in turn makes the motor move at a particular speed ($s\theta$).
- This speed is integrated to generated position (θ)
- This position is converted via a potentiometer (A) to an output voltage related to the position (y(s))
- Furthermore, this position is fed back through $N(\theta)$ which accounts for the effects of gravity that is dependent on angle

The system is nonlinear as a result of $N(\theta)$. However, this problem can be eliminated by linearising the system at a particular operating point.

Thus

$$N(\theta) = MgL\cos(\theta)$$

can be linearized at θ_0 to generate

$$\alpha = MgL\cos(\theta_0)$$

Here,

M = Mass of the robot arm system (kg)

g = gravitational constant = 9.8 m/s^2

L = Length from the pivot point to the centre of gravity

This block diagram allows the generation of a form of the plant model. This can be obtained from block diagram manipulation.

A step-by-step breakdown of the simplification (starting from the inside out) will be shown here.

$$\left[\frac{1}{s\boldsymbol{J}} \quad B\right] \Rightarrow \frac{1}{B + s\boldsymbol{J}}$$

$$\left[\frac{1}{B + s\boldsymbol{J}} \quad \frac{1}{s}\right] \Rightarrow \frac{1}{sB + s^2\boldsymbol{J}}$$

$$\frac{1}{sB + s^2\boldsymbol{J}} \quad \alpha$$

Thus the robot arm plant transfer model is reduced to a second order system with gain. The symbols in this transfer function are listed below:

A = potentiometer constant (V/rad)

K = motor constant (Nm/V)

$\boldsymbol{J}$ = inertia (Nms^2)

B = drag coefficient (Nms)

α = linearized constant (Nm)

The structure of the robot arm plant model is confirmed by the derived model for the plant that was obtained from various step input tests.

Figure 9.3 shows five open loop step results for the robot arm plant where the step size was 1.5 volts.

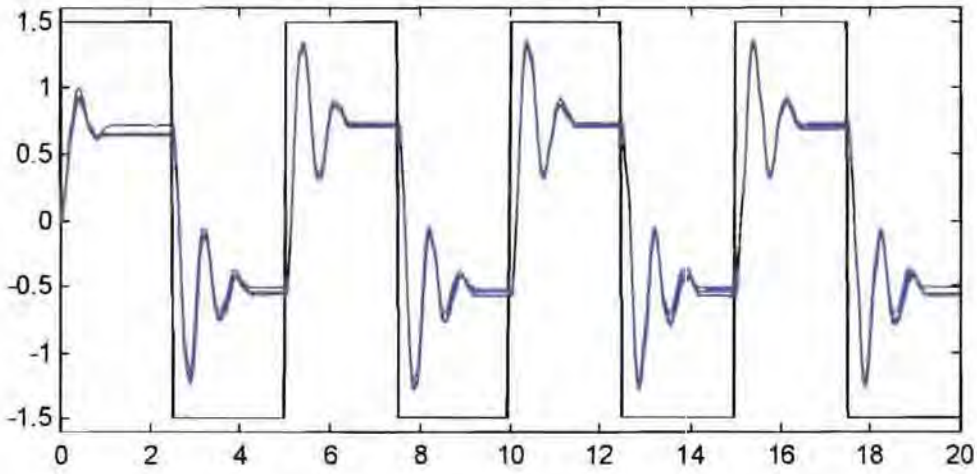


Figure 9.3 Five open loop step results for the robot arm plant with a step size of 1.5 volts

The nominal model was derived from the graph using the average value that the system settled to. The values derived from this graph are as follows:

Total average change in settling value for the first 5 seconds = 1.2381 volts for an input change of 3 volts.
There were 2 oscillations within 1.5 seconds. Therefore T=0.75 seconds.
$\omega=2\pi/T = 8.3776$.
These values yield poles at –1.4545 + j8.3776 and –1.4545 – j8.3776.

The transfer function then has the form:

$$\frac{A}{s^2 + 2.9090s + 72.2998}$$

Therefore A / 72.2998 = 1.2381 / 3. This yields A=0.4127 * 72.2998.

The final nominal model is therefore $\dfrac{0.4127}{0.0138s^2 + 0.0402s + 1}$

Another model derived from these results is $\dfrac{0.45}{0.0063s^2 + 0.0126s + 1}$

Yet another model derived from these results is $\dfrac{0.0397}{0.0213s^2 + 0.0678s + 1}$

The final complete model is then $\dfrac{0.4127 \pm 0.0373}{(0.0138 \pm 0.0075)s^2 + (0.0402 \pm 0.0276)s + 1}$

9.2.2 *Initial controller designs to optimize setpoint tracking*

In order to understand the plant better, designs were done to optimize only the sensitivity function, without taking disturbance rejection into account. The reason for these calculations was to investigate techniques for reducing the large initial input surge. As it turns out, the two examples will show that it was not any easier to minimize the input with only sensitivity minimization. The design of the robot arm plant actually makes it a difficult plant to control from an input energy perspective. Basically a large amount of input energy is required to ensure decent setpoint tracking and disturbance rejection. However this input energy is not available due to the rating of the electric motor. This could be improved in future constructions by possibly adjusting the weight on the shaft or using a different motor.

A brief description of the method follows. The reader is referred to other references for a more detailed explanation such as Feedback Control Theory or Multivariable Feedback Design (see references on page 111). The technique involves the Youla parameterization and co-prime factorizations.

Youla's parameterization using a co-prime factorization can be expressed as:

$$C(s) = \frac{X + MQ}{Y - NQ} \qquad (Q \in \text{all stable transfer functions})$$

In this factorization, the plant is represented by G=N/M, and NX+MY=1. This condition constitutes a Co-Prime factorization.

Using this parameterization, the performance spec $\|W_1 S\|_\infty < 1$ can be rewritten as $\|W_1(1 - GQ)\|_\infty < 1$.

We consider the case where the inverse of the plant is stable (ie. there are no right half plane zeroes). This is the case for the robot arm plant.

We can then set $Q = G^{-1}J$ where $J(s) = \frac{1}{(Ts+1)^k}$ ensures that Q is proper.

k is the relative degree of the plant and T is varied to ensure a satisfactory H-infinity norm.

The design for performance is then to ensure that $\|W_1 MY(1 - J)\|_\infty < 1$ by suitable choice of W_1 and T.

Once this is achieved, $Q = YN^{-1}J$ and $C = \frac{X + MQ}{Y - NQ}$.

Refer to Feedback Control Theory by John Doyle, Bruce Francis and Allen Tannenbaum for more detailed explanation of where the above derives from.

Two different designs will now be presented to see the effect of the weighting functions on the input response.

DESIGN 1

$$W_1(s) = \frac{100}{s+1}.$$

This weighting function can ensure a maximum tracking error of 1% up to about 1 rad/s.

The H-infinity norms were calculated for different values of T.

T value	H-Infinity Norm
1	**2316**
0.1	**365.5767**
0.01	**36.8675**
0.001	**3.6883**
0.0001	**0.3694**

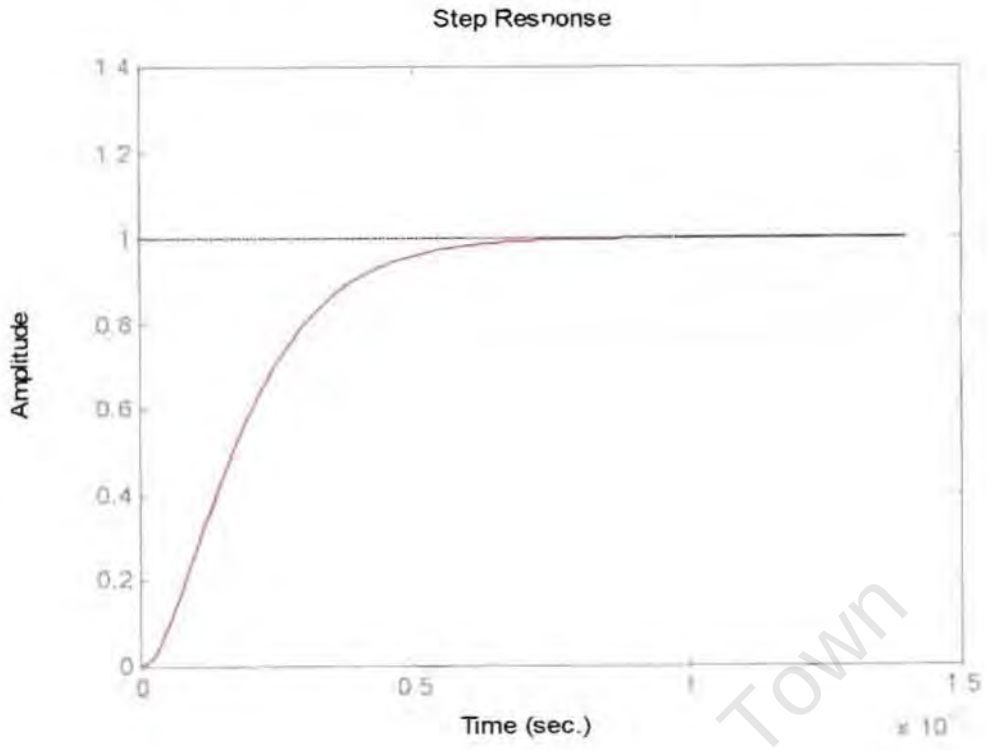
The choice is thus T=0.0001 as this ensures a norm less than 1.

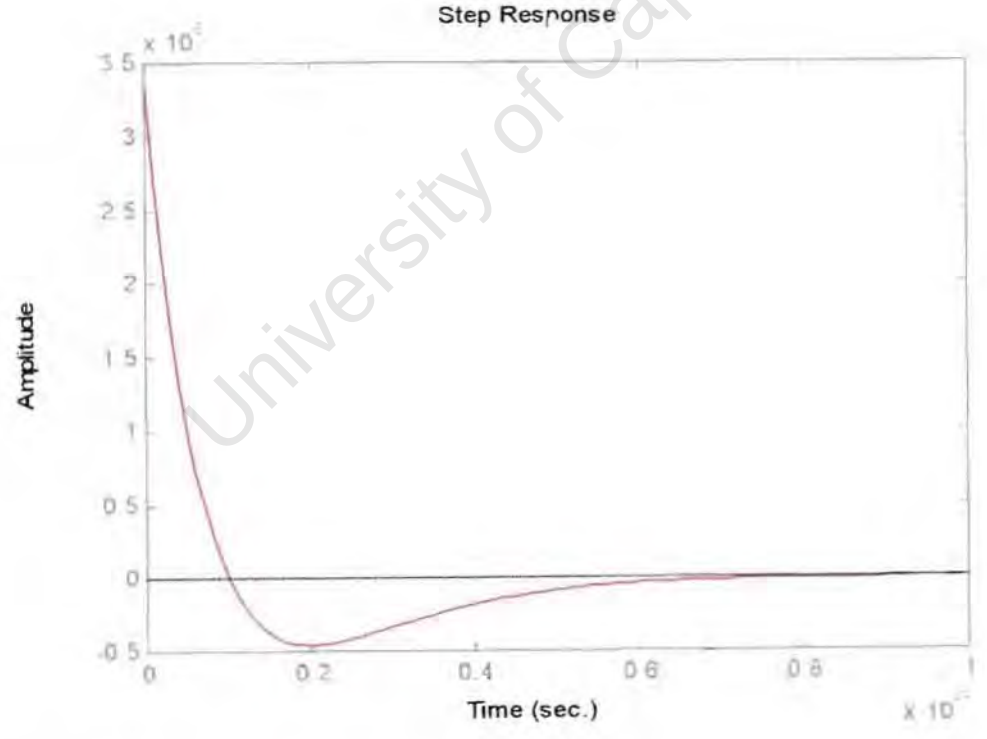
The resulting controller is given by:

$$C(s) = \frac{317400238718s^3 + 956974383174s^2 + 95266348000s + 317400000000}{94921s^3 + 1898428254s^2 + 165080000s}$$

This controller has a very fast tracking time: 0.001s with zero steady state error, due to the pole at s=0. Unfortunately the input is unrealistic. There is an initial surge of 3.5×10^6 volts. This results from the sensitivity specification being too tight. We shall thus attempt relaxing this condition in the next design.

The closed loop step response and the input response are now given.





DESIGN 2

$$W_1(s) = \frac{10}{s+1}.$$

This weighting function can ensure a maximum tracking error of 10% up to about 1 rad/s.

The H-infinity norms were calculated for different values of T.

T value	H-Infinity Norm
1	**231.6**
0.1	**36.55767**
0.01	**3.68675**
0.001	**0.36883**

The choice is thus T=0.001.

This reduces the input surge to $3.5\text{x}10^{4}$ volts. It is clear that simply reducing the gain of the weighting function will simply increase the T value that ensures a valid norm. Thus consider

$$W_1(s) = \frac{0.1}{s+1}.$$

This weighting function can ensure a maximum tracking error of 10% up to about 1 rad/s.

The H-infinity norms were calculated for different values of T.

T value	H-Infinity Norm
1	**2.316**
0.1	**0.36558**

Thus the choice is T=0.1.

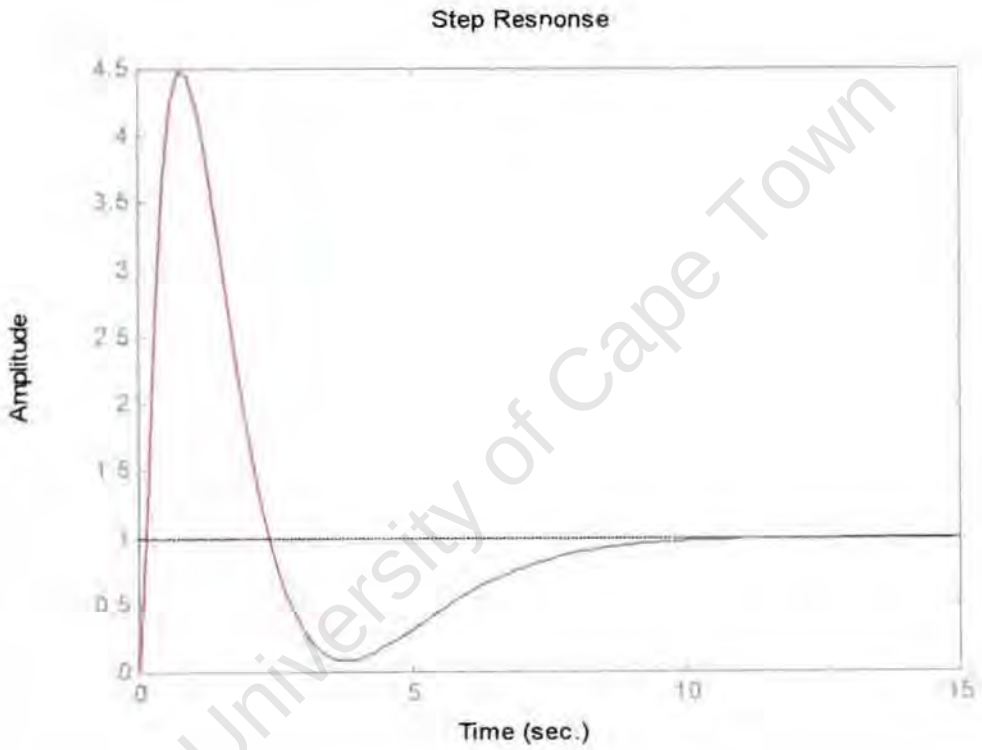
The generated controller is given by:

$$C(s) = \frac{556118s^3 + 5749734s^2 + 1415680s + 317400}{94921s^3 + 1906674s^2 + 165080s}$$

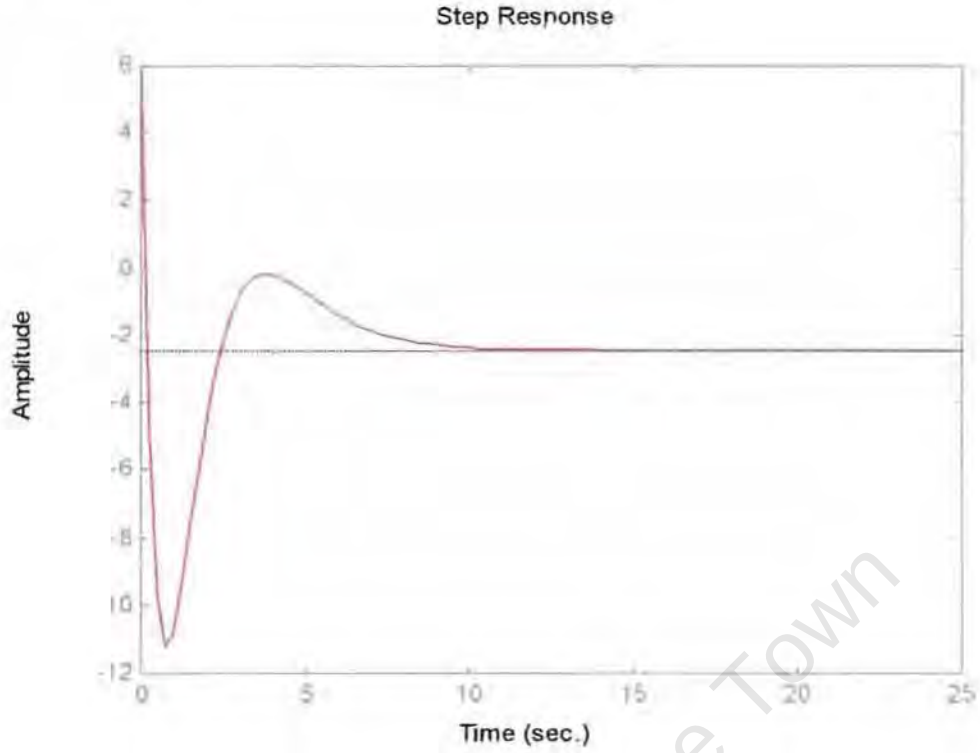
This controller has a decent input response but the output response has very large overshoot and long response time: 15s.

The plots are shown below.

Closed loop response:



Input Response:



Thus by simply reducing the gain of the filter, the input and output responses do not improve. The rest of the controller design efforts were thus concentrated on mixed performance optimization using the Matlab H-infinity routines. This yielded the final robot arm controller.

9.3 Appendix: L_2 Parametric Stability Margin

This section makes use of the following notation for set regions:

S = the region of interest

∂S = boundary of the region S

C = the complex plane

U = the area outside region S (U = C – S)

U^0 = interior of U (U^0 = C – S - ∂S)

Furthermore,

$$S \cup \partial S \cup U^0 = C \quad \text{and} \quad S \cap U^0 = S \cap \partial S = \partial S \cap U^0 = \varnothing$$

9.3.1 The Boundary Crossing Theorem

In order to present the complete proof of the Boundary Crossing Theorem, it is necessary to present a number of preliminary theorems that appear in complex variable theory.

The first fundamental theorem that will be presented without proof, is the Principle of the Argument. The reader is referred to good references on complex theory for a proof of this theorem.

Theorem 6.3.1-A (Principle of the Argument)

Let C denote a simple closed contour in the complex plane and w=f(z) a function of the complex variable z, which is analytic on C. Let Z and P denote the number of zeroes and poles, respectively, of f(z) contained in C. Let $\Delta_c$arg [f(z)] denote the net change of argument (angle) of f(z) as z traverses the contour C.

Then

$$\Delta_c \arg[f(z)] = 2\Pi(Z - P)$$

An important consequence of this theorem is the theorem of Rouche.

Theorem 6.3.1-B (Rouche's Theorem)

Let f(z) and g(z) be two functions which are analytic inside and on a simple closed contour C in the complex plane. If

$$|g(z)| < |f(z)| \quad \text{6.3.1-B.1}$$

for any z on C, then f(z) and f(z) + g(z) have the same number (multiplicity's included) of zeroes inside C.

Proof:

Since f(z) cannot vanish on C because of 6.3.1-B.1, we have

$$\Delta_c \arg[f(z)+g(z)] = \Delta_c \arg\left\{ f(z)\left[1+\frac{g(z)}{f(z)}\right]\right\}$$
$$= \Delta_c \arg[f(z)] + \Delta_c \arg\left[1+\frac{g(z)}{f(z)}\right]$$

Moreover, since

$$|g(z)| < |f(z)|$$

for all z∈ C, the variable point

$$w = 1+\frac{g(z)}{f(z)}$$

remains within the disc $|w-1|<1$ as z describes the curve C. Therefore w cannot wind around the origin which means that

$$\Delta_c \arg\left[1+\frac{g(z)}{f(z)}\right] = 0$$

Hence

$$\Delta_c \arg[f(z)+g(z)] = \Delta_c \arg[f(z)]\backslash$$

Since f(z) and g(z) are analytic in and on C, the theorem follows as an immediate consequence of the principle of the argument.

Theorem 6.3.1-C

Let

$$P(s) = p_0 + p_1 s + p_2 s^2 + \cdots + p_n s^n = \prod_{j=1}^{m} (s - s_j)^{t_j}, p_n \neq 0$$

$$Q(s) = (p_0 + \varepsilon_0) + (p_1 + \varepsilon_1)s + \cdots + (p_n + \varepsilon_n)s^n$$

and consider a circle C_k, of radius r_k, centred at s_k which is a root of P(s) of multiplicity t_k. Let r_k be fixed in such a way that

$$0 < r_k < \min |s_k - s_j| \text{ for j = 1, 2, ..., k-1, k+1, ..., m}$$

Then there exists a positive number ε such that $|\varepsilon_i| \leq \varepsilon$, for i = 0, 1, ..., n implies that Q(s) has precisely t_k zeroes inside the circle C_k.

Proof:

P(s) is non-zero and continuous on the compact set C_k and therefore it is possible to find $\delta_k > 0$ such that

$$|P(s)| \geq \delta_k > 0 \text{ for all s} \in C_k$$

On the other hand, consider the polynomial R(s), defined by

$$R(s) = \varepsilon_0 + \varepsilon_1 s + \cdots + \varepsilon_n s^n$$

If s belongs to the circle C_k, then

$$|R(s)| \le \sum_{j=0}^{n} |\varepsilon_j| \cdot |s^j| \le \sum_{j=0}^{n} |\varepsilon_j| \cdot (|s - s_k| + |s_k|)^j$$

$$\le \varepsilon \sum_{j=0}^{n} (r_k + |s_k|)^j$$

$$M_k$$

Thus if ε is chosen so that $\varepsilon < \frac{\delta_k}{M_k}$ it is concluded that

$|R(s)| < |P(s)|$ for all s on C_k

so that by Rouche's Theorem P(s) and Q(s) = P(s) + R(s) have the same number of zeroes inside C_k. Since the choice of r_k ensures that P(s) has just one zero of multiplicity t_k at s_k, it can be seen that Q(s) has precisely t_k zeroes in C_k.

Corollary

Fix m circles C_1, ..., C_m, that are pairwise disjoint and centered at s_1, s_2, ..., s_m respectively. By repeatedly applying the previous theorem, it is always possible to find an $\varepsilon > 0$ such that for any set of numbers $\{\varepsilon_0, ..., \varepsilon_n\}$ satisfying $|\varepsilon_i| \le \varepsilon$ for i = 0, 1, ..., n, Q(s) has precisely t_j zeroes inside each of the circles C_j.

The above theorems now lead to the Boundary Crossing Theorem.

Theorem 6.3.1-D (The Boundary Crossing Theorem)

Consider a family of polynomials

$$P(\lambda, s) = p_0(\lambda) + p_1(\lambda) \cdot s + p_2(\lambda) \cdot s^2 + \ldots + p_n(\lambda) \cdot s^n$$

The following observations can be made about this family:

- The family is of fixed degree n (invariant degree)
- The family is continuous with respect to λ on a fixed interval I = [a,b]

Furthermore, $p_0(\lambda), p_1(\lambda), \ldots, p_n(\lambda)$ are continuous functions of λ on I and $p_n(\lambda) \neq 0$ for all $\lambda \in I$. From theorem 6.3.1-C it is immediate in general, that for any open set O, the set of polynomials of degree n that have all their roots in O, is itself open.

Under these assumptions the following theorem can be stated:

Theorem (Boundary Crossing Theorem)

Suppose that P(a,s) has all its roots in some region S, whereas P(b,s) has at least one root in region U. Then there exists at least one ρ in (a,b] such that:

- P(ρ,s) has all its roots in $S \cup \partial S$
- P(ρ,s) has at least one root in ∂S

Where ∂S denotes the boundary of the region S

Proof

To prove this result, introduce the set E of all real numbers t belonging to (a,b] and satisfying the following property:

P(t',s) has all its roots in region S, for all t' $\in$ (a,t)

By assumption, P(a,s) itself has all its roots in region S and therefore, it is possible to find an $\alpha > 0$ such that

For all $t' \in [a, a + \alpha) \cap I$, P(t',s) also has all its roots in region S

From this, it can be concluded that E is not empty since, for example $a + 0.5 * \alpha$ belongs to E.

Moreover, from the definition of E the following property holds:

$t_2 \in E$ and $a < t_1 < t_2$ implies that t_1 itself belongs to E.

Given this, it is clear that E is an interval and if

$$\rho := \sup_{t \in E} t \qquad \text{6.3.1-D.1}$$

then it is concluded that $E = (a,\rho]$.

A)

On the one hand it is impossible that $P(\rho,s)$ has all its roots in region S. If this were the case then necessarily $\rho < b$ and it would be possible to find an $\alpha > 0$ such that $\rho + \alpha < b$ and

For all $t' \in (\rho - \alpha, \rho + \alpha) \cap I$, P(t',s) also has all its roots in region S

As a result, $\rho + \frac{\alpha}{2}$ would belong to E and this would contradict the definition of ρ in 6.3.1-D.1.

B)

On the other hand, it is impossible that P(ρ,s) has even one root in the interior of U, because by applying theorem 6.3.1-C grants the possibility of finding an $\alpha > 0$ such that

For all t' $\in (\rho - \alpha, \rho + \alpha) \cap I$, P(t',s) also has at least one root in U^0,

and this would contradict the fact that $\rho - \varepsilon$, belongs to E for ε small enough.

From A) and B) it is thus concluded that P(ρ,s) has all its roots in $S \cup \partial S$ and at least one root in ∂S.

9.3.2 *Parametric Stability Margin*

Let S denote the stability region of interest where $S \subset C$.

Let **P** be a vector of real parameters:

$$\mathbf{p} = [p_1, p_2, \ldots, p_l]^T$$

The characteristic polynomial for the closed loop system can then be denoted by:

$$\delta(s, \mathbf{p}) = \delta_n(\mathbf{p}) \cdot s^n - \delta_{n-1}(\mathbf{p}) \cdot s^{n-1} + \ldots + \delta_1(\mathbf{p}) \cdot s^1 + \delta_0(\mathbf{p}) \cdot s^0$$

This polynomial is therefore a real polynomial with coefficients that depend continuously on the real parameter vector **p**. For the nominal parameter $\mathbf{p}=\mathbf{p}^0$, $\delta(s, \mathbf{p}^0) = \delta^0(s)$ is stable with respect to region S (all the roots lie in S).

The perturbation in the parameter **p** from its nominal value $\mathbf{p}^0$ can be denoted by:

$$\Delta\mathbf{p} = \mathbf{p} - \mathbf{p}^0 = [p_1 - p_1^0, p_2 - p_2^0, p_3 - p_3^0, \ldots, p_l - p_l^0]$$

Now if a norm is introduced in the space of the parameters **p**, an open ball of radius ρ can be introduced:

$$\beta(\rho, \mathbf{p}^0) = \{\mathbf{p} : \|\mathbf{p} - \mathbf{p}^0\| < \rho\}$$

The hypersphere of radius ρ can be defined by

$$S(\rho, \mathbf{p}^0) = \{\mathbf{p} : \|\mathbf{p} - \mathbf{p}^0\| = \rho\}$$

With the ball β we associate the family of uncertain polynomials

$$\Delta_\rho(s) := \{\delta(s, \mathbf{p}^0 + \Delta\mathbf{p}) : \|\Delta\mathbf{p}\| < \rho\}$$

The real parametric stability margin in parameter space can then be defined as the radius, denoted $\rho^*(\mathbf{p}^0)$ of the largest ball centered at $\mathbf{p}^0$ for which $\delta(s, \mathbf{p})$ remains stable whenever $p \in \beta(\rho^*(\mathbf{p}^0), \mathbf{p}^0)$.

Theorem (Parametric Stability Margin):

a) There exists a largest stability ball $\beta(\rho^*, \mathbf{p}^0)$ centered at $\mathbf{p}^0$ with the property that:

1. For every **p'** within the ball, the characteristic polynomial δ(s, **p'**) is stable and of degree n

2. At least one point **p''** on the hypersphere S(ρ*,**p**^{0}) itself is such that δ(s, **p''**) is unstable or of degree less than n.

b) Moreover if **p''** is any point on the hypersphere S(ρ*,**p**^{0}) such that δ(s, **p''**) is unstable, then the unstable roots of δ(s, **p''**) can only be on the stability boundary.

Property 7.3.2.1

$$\left\|\psi(s,\mathbf{p}) - \delta(s,\mathbf{p}^0)\right\| < \varepsilon \quad \text{where the degree of } \psi(s,\mathbf{p}) = \text{n and } \psi(s,\mathbf{p}) \text{ has all its roots in S}$$

Given a stable polynomial $\delta(s,\mathbf{p}^0)$, the subset of all real numbers having property 7.3.2.1 is given by:

$$R_\delta := \{t : t > 0, t \text{ satisfies property } 7.3.2.1$$

Now R_δ is not empty since if $t_2 \in R_\delta$ and $0 < t_1 < t_2$, then $t_1 \in R_\delta$

Therefore R_δ is an interval $(0,\rho(\delta)]$ where $\rho(\delta) = \sup_{t \in R_\delta}(t)$

Therefore ρ(δ) satisfies property 7.3.2.1.

Proof

a1) is true since ρ^* satisfies property 7.3.2.1

a2 and b)

Since no real $r > \rho^*$ satisfies property 7.3.2.1, then for every n≥1, there exists a polynomial of degree less than n or with a root in U=C-S, say $\gamma(s,\mathbf{p}^\gamma)$ contained in the ball $\beta\left(\rho^* + \frac{1}{n}, \delta(s,\mathbf{p}^0)\right)$. Being contained in the closure of $\beta(\rho^* + 1, \delta(s,\mathbf{p}^0))$, which is a compact set, this sequence must contain a convergent subsequence $\gamma_{\phi n}(s,\mathbf{p}^{\gamma^*})$. Let $\gamma(s,\mathbf{p}^\gamma)$ be its limit. Then $\gamma(s,\mathbf{p}^\gamma)$ is lying on the hypersphere $S(\rho^*, \delta(s,\mathbf{p}^0))$ and is of degree less than n or with a root in U, otherwise the existence of $\rho^*(\mathbf{p}^\gamma)$ would contradict the fact that $\gamma(s,\mathbf{p}^\gamma)$ is the limit of a sequence of polynomials of degree less than n or with a root in U.

Invoke Rouche's Theorem. Suppose that there is a polynomial lying on $S(\rho^*, \delta(s, \mathbf{p}^0))$, say $\gamma(s, \mathbf{p}^\gamma)$, which is of degree n but has at least one root s_k in U^0. A consequence is that the set of polynomials of degree n with a least one root in the open set U^0 is itself open. It would then be possible to find a ball of radius $\varepsilon > 0$ around $\gamma(s, \mathbf{p}^\gamma)$ containing only polynomials of degree n with at least one root in U^0. This would result in a contradiction because since $\gamma(s, \mathbf{p}^\gamma)$ lies on the hypersphere $S(\rho^*, \delta(s, \mathbf{p}^0))$, the intersection $\beta(\rho^*, \delta(s, \mathbf{p}^0)) \cap \beta(\varepsilon, \gamma(s, \mathbf{p}^\gamma))$ is not empty.

On the other hand, suppose that $\gamma(s, \mathbf{p}^\gamma)$ with at least one root in U^0 is of degree less than n. For $\varepsilon>0$ consider the polynomial $\gamma_\varepsilon(s, \mathbf{p}^{\gamma\varepsilon}) = \varepsilon\delta(s, p^0) + (1-\varepsilon)\gamma(s, \mathbf{p}^\gamma)$. It is clear that $\gamma_\varepsilon(s, \mathbf{p}^{\gamma\varepsilon})$ is always of degree n and is inside $\beta(\rho^*, \delta(s, \mathbf{p}^0))$ since

$$\left\| \delta(s, \mathbf{p}^0) - \gamma_\varepsilon(s, \mathbf{p}^{\gamma\varepsilon}) \right\| = (1-\varepsilon) \left\| \delta(s, \mathbf{p}^0) - \gamma(s, \mathbf{p}^\gamma) \right\| < \rho^*.$$

This means that $\gamma_\varepsilon(s, \mathbf{p}^{\gamma\varepsilon})$ has all its roots in S. By applying Rouche's Theorem as $\varepsilon \to 0$, $\gamma_\varepsilon(s, \mathbf{p}^{\gamma\varepsilon})$ has at least one root in U^0 and this is also a contradiction.

9.4 Appendix: Robot Arm Digital Controller Software

This section provides an overview of the software that was used to control the robot arm plant digitally. The program was written using Microsoft Visual C++ 5.0 and was compiled to run under windows NT 4.0.

9.4.1 Features and limitations

An overview of the features are listed below:

- Simulation of a system using state space models
- Physical control of a system using ADC/DAC cards under NT
- Discrete time and continuous time simulation of a system
- Open and closed loop simulation is possible
- Input limiting is included
- Direct import of models from a matlab export. The matlab routines to generate the model import files are included.
- Nonlinear simulation is also included but is limited to a fixed plant model.
- Both the stable and unstable plant models can be simulated under one model file. This allows the user to observe the responses in both regions for a particular controller.

The limitations of the current version are listed below:

- Model entry is limited to state space notation
- Only the plant model and controller model can be customised. Unity feedback has been assumed.
- Only step input functions have been created. The random waveform is not completely random, but it can be customised in terms of successive step sizes and the length of each step.
- No allowance has been made for dead times. This is not a major issue in the robot arm plant and has thus been ignored.
- No printing or on-line help facilities are included.

9.4.2 *Code implementation overview*

A simplified class structure is shown in Figure 9.4. It shows the main interactions between the classes that form the engine of the robotarm software.

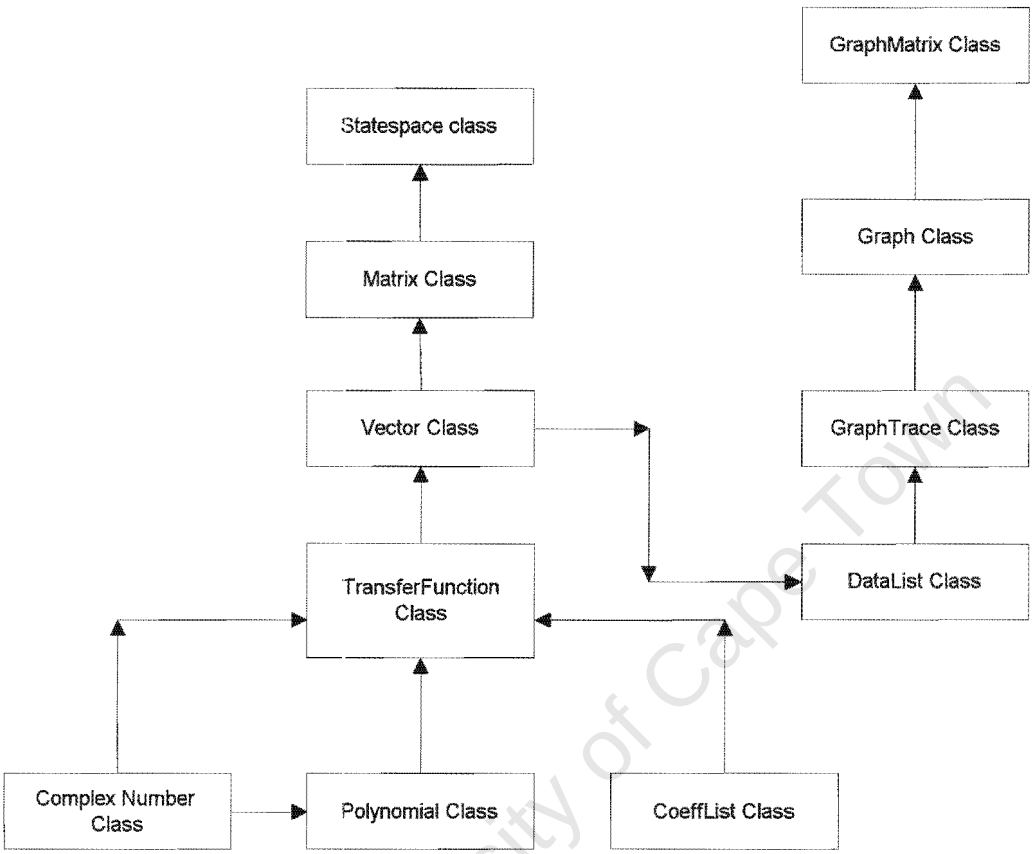


Figure 9.4 Simplified class hierarchy showing the main class interactions.

The arrows point to the class that uses the child. Thus the polynomial class uses the complex number class.

Complex Number Class

The complex number class provides support for complex numbers. This is used by the graph plotting class to form data points, where the real part is the horizontal axis value and the imaginary part is the vertical axis value. It provides functions for most arithmetic operations, including power operators. It has no class dependencies as it is formed entirely using real numbers. C++ has support for floats, doubles and long doubles.

The data type used for storing numbers in the RobotArm software is primarily the double data type that is an 8 byte number that ranges from –1.7e±308 to 1.7e±308. This data type has 15 digit precision which is sufficient accuracy for this application.

CoeffList Class

This class stores the coefficients of polynomials. These coefficients are read in by parsing a string expression for the polynomial. In the Robotarm software, this was only used for implementation of the non-linear plant model.

Polynomial Class

This class implements a polynomial in the variable 's'. The coefficients are stored in a CoeffList structure. This class provides routines to multiply, divide, add and subtract polynomials. There are also routines to evaluate polynomials at complex number values and to convert polynomials to string representations.

Transfer Function Class

This class implements a transfer function in 's'. The numerator and denominator are polynomials. This class implements arithmetic functions for adding, subtracting, multiplying and dividing transfer functions. This class also holds the code for implementing a Runge Kutta simulation of the transfer function. The simulation code is only for continuous time simulation. No discrete simulation is included for the transfer function models.

DataList Class

This class stores vectors of data elements. It was designed to store complex number vectors. These complex numbers are typically data points for the graph plots, although they could be any other data type. This class provides routines to add and delete elements, and also to copy data lists from one structure to the next.

GraphTrace Class

This class uses the DataList class to store the data points for a particular graph trace. It also stores the color of the trace for plotting purposes.

Graph Class

This class implements a cartesian graph with an associated trace – a GraphTrace structure. Each graph can have many traces and the graphs provide automatic rescaling based on their axes. There are routines for plotting traces and trace labels. Zooming facilities are provided for so that one can zoom in or out of a graph up to 4 times. Each graph has its own printing facilities to ensure a decent sized printout. This is achieved by scaling the graph's output when printing is selected.

GraphMatrix Class

This class implements a matrix of graph plots to create a grid of plots. It allows for selection of plots, maximization and minimization of selected plots and automatic scaling of each plot to the window. It stores a number of Graph objects, one for each plot on the grid.

Vector Class

This class implements a general vector. The data type stored in the vector can be any known type, such as transfer functions, polynomials, doubles, integers, complex numbers, etc. This class provides such functions as dot products, element extraction, element exclusion and vector evaluation: where a transfer function vector is evaluated at a particular frequency. This class supports both column and row vectors. The transpose routine can be used to switch between one and the other.

Matrix Class

This implements a general matrix class. The elements of the matrix can be any known type as in the vector case. Besides the normal arithmetic functions, routines are also provided to invert complex matrices, calculate determinants and evaluate transfer function matrices at different frequencies. The matrix inversion is achieved using Gauss-Jordan elimination with full pivoting. The matrix is used for storing the state space representations of the different models.

Statespace Class

This implements a state space model of a plant or controller. The most important routines that are implemented here are the simulation routines for continuous and discrete time. The discrete time simulation is achieved by addition, subtraction and multiplication of vectors and matrices. The continuous time simulation is an extension of the simulation code in the transfer function class. It performs Runge Kutta order 4 using matrices as its input.

Conversion Routines

This is not shown on the class structure as it is not a class. However, this file contains routines that are used in the graph and polynomial classes. It has routines to convert double numbers to strings for output. It also allows selection of the number of decimal places to show in the string output.

Error Handler

This is also not shown on the class structure. This routine outputs an error message and terminates the program if the error is fatal, for example, when access is made to invalid memory locations.

These are the main classes that make up the robot arm control software. All the mathematical code was written and tested by comparison with results from mathematical packages such as Mathcad v6.0 and Matlab v5.0.

9.4.3 *Mathematical background to digital simulation*

The digital simulation in the robotarm software is based on Runge Kutta Order 4 – a method that is widely used for approximating differential equations. There are two cases for this simulation: the state space simulation for the models entered in the program and the simulation of a transfer function as in the case of the non-linear model for the robot arm plant. The differential equations for the transfer function can be obtained by means of state space transforms.

Consider the process $G(s) = \dfrac{a_m \cdot s^m + a_{m-1} \cdot s^{m-1} + \ldots + a_1 \cdot s + a_0}{b_n \cdot s^n + b_{n-1} \cdot s^{n-1} + \ldots + b_1 \cdot s + b_0} = \dfrac{y}{u}$

where u is the input to G(s)

y is the output of G(s)

This transfer function can be converted to state space by multiplying by $\dfrac{w}{w}$

Equating denominators yields :

$$u = b_n \cdot s^n w + b_{n-1} \cdot s^{n-1} w + \ldots + b_1 \cdot sw + b_0 \cdot w$$

Rearranging this equation yields :

$$s^n w = -\frac{b_{n-1}}{b_n} \cdot s^{n-1} w - \ldots - \frac{b_1}{b_n} \cdot sw - \frac{b_0}{b_n} \cdot w + \frac{u}{b_n}$$

Assign the state vector **x** to represent the states in the system :

$$x_0 = w$$

$$x_1 = sx_0 = s^1 w$$

$$x_2 = sx_1 = s^2 w$$

$$\vdots$$

$$x_{n-1} = sx_{n-2} = s^{n-1} w$$

Consider the case where $m = n - 1$

Then $y = a_m \cdot x_{n-1} + a_{m-1} \cdot x_{n-2} + \ldots + a_1 \cdot x_1 + a_0 \cdot x_0$

Runge Kutta Order 4 is applied to each of the 1st order state differenti al equations to solve for the states of **x**.
Initially the state vector is set to 0. The states are then used recursivel y to solve for further output values.

Note that if m = n then the y output value will depend on the states as well as the n' th derivative, ie : y will include a term $a_n \cdot s^n w$ which cannot be expressed in terms of the states. This implies that the output is using a current value instead of previous values only. Stated in another way : if we divided the denominator into the numerator we would have a constant plus a transfer function with m < n. The constant allows current values to pass through the simulation step instantaneously causing spurious results.

For each transfer function there are 2 mappings: an input to state mapping and a state to output mapping. To prevent values from passing through a transfer function instantaneously impose the restriction m<n for the process matrix. The simulation then begins at the output mapping of G(s) and proceeds around the feedback loop first calculating the entire input mapping, and then calculating the output mapping using the states. The simulation step ends once the input to state mapping for the process G(s) is completed. This process is illustrated in Figure 9.5 below.

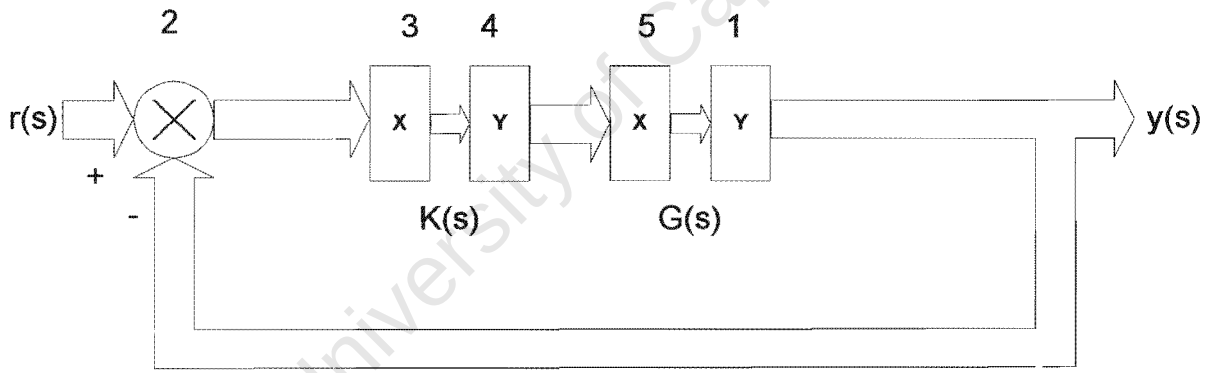


Figure 9.6: The order of evaluation of mappings for digital simulation

X represents the state vector for the transfer function and **Y** represents the output vector for the transfer function.

In the case of the state space simulation, the route followed is very similar to that of the transfer function. The major difference is that the conversion from transfer function to state space need not be done. The number of states is calculated from the number of columns in the process or controller state space matrix. Each successive step in simulation is then directly calculated from $\dot{x} = \mathbf{A}x + \mathbf{b}u$ and $\mathrm{y} = \mathbf{C}\mathrm{x}^{\mathrm{T}} + du$.

9.4.4 *Abridged User's Manual for the robot arm software*

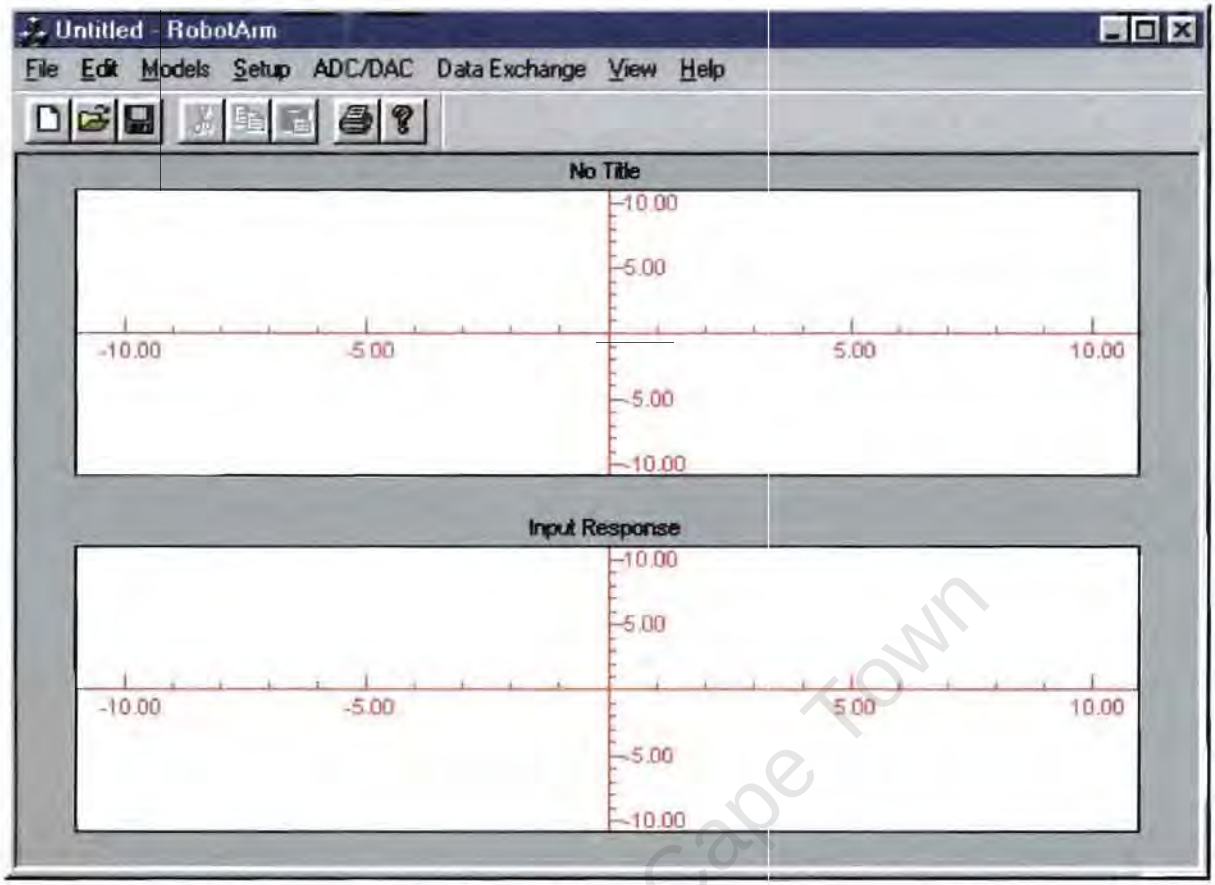


Figure 9.6 The main robot arm program screen

The menus in the robot arm software

The main screen includes:

- an output response graph capable of showing the output, the setpoint and the error between the output and the setpoint
- an input response graph capable of showing the input response to the plant

These graphs can be customised via the **setup** menu.

The Models menu

The Models menu contains five functions:

- Plant Model Stable: This is used to enter the linearised model for the plant in the stable region
- Plant Model Unstable: This is used to enter the linearised model for the plant in the unstable region
- Controller Model: This is used to enter the model for the controller
- Simulate: This launches the simulator for discrete time, continuous time or non-linear simulation

- Set Active Model: This switches between the stable and unstable models for simulation in the case of linear simulation

The ADC/DAC Menu

This menu only has one option.

- Simulate Physical System: This applies the specified controller to the physical plant by means of digital-to-analog and analog-to-digital converters. No output is plotted while the simulation runs. This was done to speed up the computation of the control loop. Once the physical perturbing of the system is complete, the output graph is plotted.

The Data Exchange Menu

This menu has one option.

- Import Model File: This imports a model file export from matlab. The export file contains one controller and one model. The plant model that is adjusted depends on which model has been set active via the models menu. The routine to create this model file has been supplied. It is mentioned under the matlab routines in section 9.4.5. The model file should be titled matlab.log and reside in the same directory as the control software.

Model entry dialog

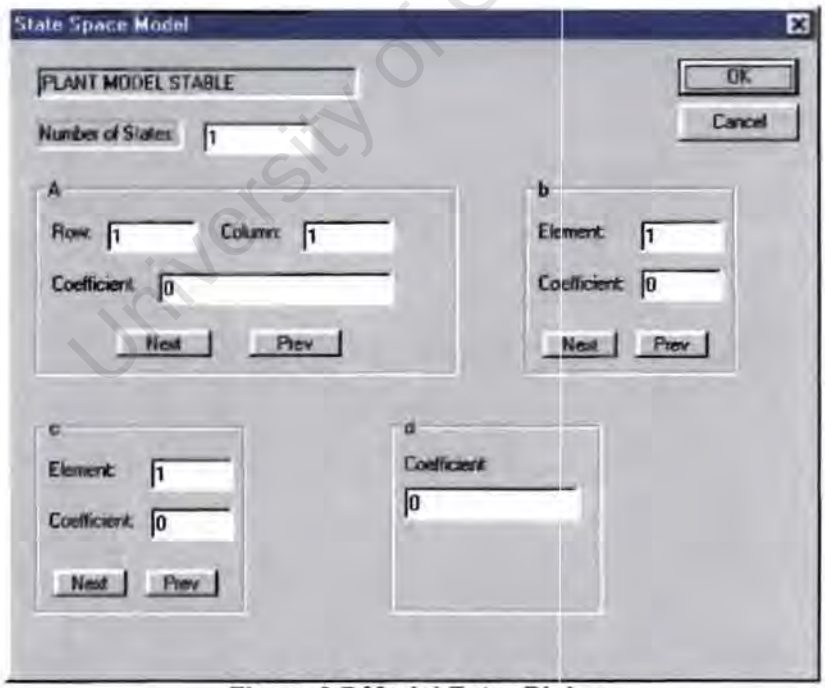


Figure 9.7 Model Entry Dialog

The model entry screen is rather self-explanatory. Models can only be entered in state space form. The four components **A**, **B**, **C**, d of a state space model can be entered value for value. The Number of States determines the size of the matrices and vectors.

Signal Generator Dialog

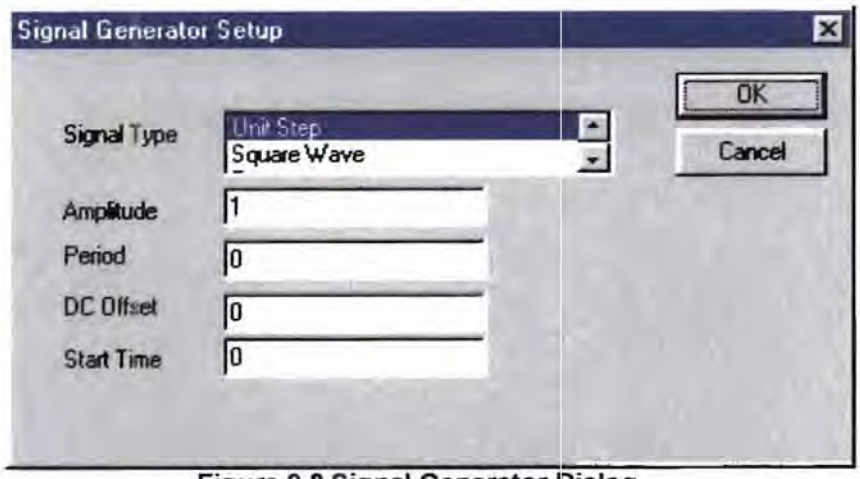


Figure 9.8 Signal Generator Dialog

The signal type determines the shape of the input signal. The only signals that have been implemented are step based: the Unit Step, the Square Wave and the Random wave.

The characteristics are self-explanatory for the Unit Step and the Square Wave. The Random wave, however, needs elaboration. The amplitude for the random wave determines the step size from one transition to another. The period determines how long each transition lasts. If the period is omitted, the random wave will not function properly. For example, a set of parameters used for physical perturbation of the real system was: amplitude = 1, period = 3, DC Offset = 0, start time = 0.

Digital Simulation Dialog

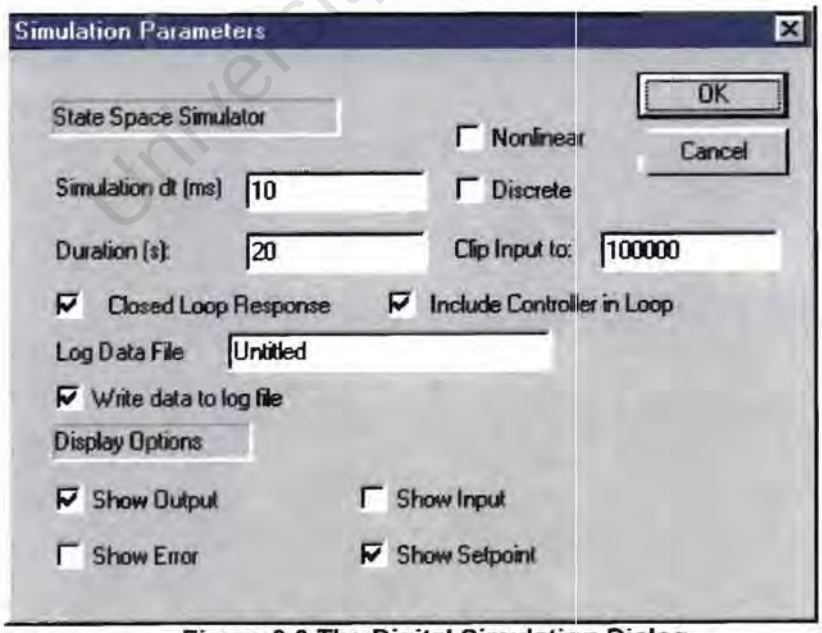


Figure 9.9 The Digital Simulation Dialog

The parameters on the digital simulation dialog are listed below, together with a brief description:

- Simulation dt (ms): This is the step size in millisecond units. Therefore, 1 millisecond will be entered as 1.
- Duration: This is the length of the simulation in seconds
- Clip input to: This is the maximum limit for the input. For example if the input limit is set at 10 volts, then if the input goes above 10 volts it is set to 10 volts and if it goes below -10 volts, it is set to –10 volts.
- Nonlinear: This determines whether the continuous non-linear model is used or if linear simulation is performed. The non-linear model cannot be customised and is fixed as the nominal model of the robot arm plant
- Discrete: This determines whether discrete or continuous time simulation is performed. If discrete is chosen the controller and plant models must be discrete time models.
- Closed loop response: This determines whether open loop or closed loop simulation is performed
- Include Controller in Loop: This determines whether the controller is included in the control loop.
- Log Data File: This specifies the filename of the log file to which the graph data will be written.
- Write Data to Log File: This determines whether the simulation data is logged or not.
- The display options determines which graph traces to display.

If the non-linear simulation is selected together with the discrete option, then the simulation will actually used a continuous time non-linear model with a discrete time controller excluding a zero-order-hold circuit in the plant. If non-linear is not selected, then the plant models for the stable and unstable regions should be discrete time models.

Real Time Simulator Dialog

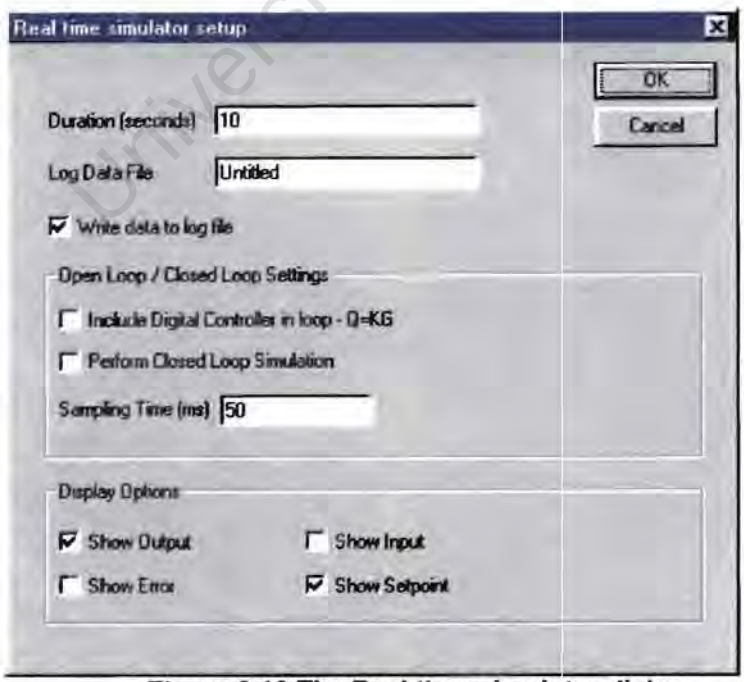


Figure 9.10 The Real time simulator dialog

The options under real time simulation are similar to those of the digital simulation, but more limited.

- Duration: This is the length of the real time perturbing in seconds
- Log Data File: This is the name of the file to which the real time data will be logged
- Write data to log file: This determines whether the data is logged or not
- Include Digital Controller in Loop: This determines whether the controller is included or not.
- Perform closed loop simulation: This determines whether open or closed loop control should be performed
- Sampling Time: This is the length of time between samples in milliseconds. The loop has been designed for 1 millisecond. Any other sampling time will not be correctly reflected as the delay is fixed at 1 millisecond.
- The display options determines what graph traces to show. This was retained for similarity with the digital simulation. However, due to the high sampling speed required, the only data that is logged and displayed is the output and the setpoint, irrespective of these options.

9.5 Appendix: Custom Matlab Routines

The section lists a number of custom matlab routines written for the robotarm plant. These routines make use of the Robust Control Toolbox and the Symbolic Toolbox that is available for Matlab. The routines were designed to run under Matlab v5.0. Not all the routines written have been included, but rather an abridged list.

9.5.1 Function List

function [n,d]=GenFilter(DCGain,HFGain,CrossOver,Damp1,Damp2)

This generates a second order filter with the specified DC Gain, High frequency gain, crossover frequency, and damping factors. This function was used to generate the desired sensitivity and complementary sensitivity functions for H-infinity design.

function [Sn,Sd,Tn,Td]=GenS_T(Pn,Pd,Kn,Kd)

This function calculates the sensitivity and complementary sensitivity functions for a given plant and controller.

function [Yout,Uout,Eout]=ManualSim(Pnz,Pdz,Knz,Kdz,Input)

This function performs a discrete time simulation of a plant and controller using a specified input graph. The sample time is determined from the plant and controller models. Only Z-transformed models will work. This version uses transfer function representation for the models.

function [Yout,Uout,Eout]=ManualSimSS(Ap,Bp,Cp,Dp,Ak,Bk,Ck,Dk,Input,InputLimit)

This function performs a discrete time simulation of a plant and controller using state space models. This version also includes input limiting.

function [fHandle]=OpenDataFile(Filename)

This function is used to open a data file for exporting models. This exported file is then used in the robot arm control software.

function [P]=Pick(DataPoints)

This function computes the pick matrix as defined in the Nevanlinna-Pick algorithm for solving H-infinity problems.

function [n,d]=PlotFilter(DCGain,HFGain,CrossOver,Damp1,Damp2,ClearP)

This function shows a bode plot of a filter.

function [Output,Input,Time]=ReadLog(Filename);

This function will read a logged data file from the robot arm control software into an array in Matlab. This is used for re-plotting graphs from simulation.

function [realpoly,imagpoly]=RealImag(inputpoly)

This function splits a polynomial into real and imaginary parts. This is used as an interim step to calculating the parametric stability margin of a polynomial.

function [Pn,Pd,Aq,Bq,Cq,Dq,Ah,Bh,Ch,Dh,Ak,Bk,Ck,Dk,As,Bs,Cs,Ds,Au,Bu,Cu,Du]=
RobotArm(Sn,Sd,Tn,Td,Un,Ud,Log,LogF);

This function performs the H-infinity control for the robot arm plant, given a desired sensitivity, complementary sensitivity and input energy specification. The routine returns the generated controller, the sensitivity function, the complementary sensitivity function and the open loop state space model.

function [dCn,dCd,dPoles]=Robustness(Pn,Pd,Kn,Kd,maxPercent)

This function calculates the robustness of a system when a certain maximum perturbation is applied to different coefficients of the controller. The limit here is that the percentage should be smaller that 100%. This is realistic as a real controller should not deviate by large amounts from the designed controller when implemented. These tests were performed on the robot arm controller and found to be stable for all perturbations less than 100% of the original coefficient.

function [Pn,Pd,Aq,Bq,Cq,Dq,Ah,Bh,Ch,Dh,Ak,Bk,Ck,Dk,As,Bs,Cs,Ds,Au,Bu,Cu,Du]=TestPlant1

This function performs H-infinity control for the test plant of 6.1.5.

function [Result,ClosedLoop]=Unstable(Pn,Pd,Cn,Cd,Cnp,Cdp)

This function determines whether the closed loop system becomes unstable when a perturbation contained in [Cnp,Cdp] is added to the original controller [Cn,Cd]

function WriteLog(LogF,Ak,Bk,Ck,Dk,ag,bg,cg,dg)

This function writes a log file contained the plant and controller models. This file is used as the import file for the robot arm control software.

function [freqr,rhowr,perturbsr]=robotal2(Sfreq,Efreq,Stepfreq)

This function calculates the L_2 parametric stability margin for the robot arm plant using symbolic mathematical routines.

function [freqr,rhowr,perturbsr]=TestPAL25(Sfreq,Efreq,Stepfreq)

This function calculates the L_2 parametric stability margin for the test plant (section 6.1.5) using symbolic mathematical routines.

function [freqr,rhowr,perturbsr]=eg1(Sfreq,Efreq,Stepfreq)

This function calculates the L_2 parametric stability margin for example 1 in the paper of Keel et al (1997) using symbolic mathematical routines.

function [freqr,rhowr,perturbsr]=eg2(Sfreq,Efreq,Stepfreq)

This function calculates the L_2 parametric stability margin for example 3 in the paper of Keel et al (1997) using symbolic mathematical routines.

9.6 Appendix: Additional Electronic Aids

An excel spreadsheet with embedded code has been included that allows one to investigate different perturbation vectors, the resulting parametric stability margin and the closed loop poles for the perturbed closed loop system. The spreadsheet also calculates the relative size of the perturbation to the original coefficient.

The code for root evaluation is an eigenvalue method based on a companion matrix in Hessenberg form.

Four files are included for: the robot arm plant, the modified robot arm plant, example 1 in the paper by Keel et al (1997) and example 3 in the paper by Keel et al (1997). These files are designed for Microsoft Office 97.

9.7 References

Bhattacharyya S.P., Chapellat H., Keel L.H.	Robust Control: The Parametric Approach, Prentice Hall, 1995
Bhattacharyya S.P., Keel L.H.	"Robust, Fragile, or Optimal?", IEEE Transactions on Automatic Control, Vol 42 No. 8, August 1997
Braae M.	A robot arm for a first course in control engineering, IEEE Transactions on education, Vol 39 No. 1, February 1996
Braae M.	Control Theory for Electrical Engineers, UCT Press, 1994
Braae M.	Digital and State Space Control Theory, UCT Press, 1996
Doyle J.C., Francis B.A., Tannenbaum A.R.	Feedback Control Theory, Macmillan, 1992
Francis B.A.	A Course in Linear H-infinity Control Theory, Springer-Verlag, 1987
Maciejowski J.M.	Multivariable Feedback Control, Addison Wesley, 1989